시계열 데이터베이스를 위한 서브시퀀스 매칭 후처리 과정의 최적화

김 상욱*, 박 대현*, 이 헌길*, 정 병대**, 손 성용**

강원대학교 컴퓨터정보통신공학부*

포디홈네트(주)**

Optimizing the Post-Processing Step of Subsequence Matching in Time-Series Databases

Sang-Wook Kim*, Dae-Hyun Park*, Heon-Gil Lee*, Byong-Dae Jung**, Sung-Yong Son**

Division of Computer, Information, and Communications Engineering Kangwon National University*

4DHomeNet Inc.**

요약

본 논문에서는 시계열 데이터베이스에서 서브시퀀스 매칭을 효과적으로 처리하는 방안에 관하여 논의한다. 먼저, 서브시퀀스 매칭의 후처리 과정에서 발생하는 기존 기법의 문제점을 지적하고, 이를 해결할 수 있는 최적의 기법을 제안하였다. 제안된 기법은 이진 트리 내에 후보 시퀀스에 대한 정보를 삽입해 둠으로써 같은 시퀀스에 속하는 후보 윈도우들과 같은 서브시퀀스에 속하는 후보 윈도우들을 연속적으로 처리하는 방식을 사용한다. 이 결과, 디스크 액세스와 서브시퀀스 비교의 측면에서 중복 작업을 완전히 제거할 수 있다. 제안된 기법의 성능 개선 효과를 검증하기 위하여 실제 주식 데이터를 위한 성능 평가를 수행하였다. 실험 결과에 의하면, 제안된 기법은 기존의 기법과 비교하여 전체적으로 55배에서 156배까지의 성능 개선 효과가 있는 것으로 나타났다.

1. 서론[1)]

시계열 데이터베이스(time-series databases)란 각 객체의 변화되는 값들의 연속으로 구성된 데이터 시퀀스(data sequences)들의 집합이며, 유사 검색(similarity search)이란 주어진 질의 시퀀스(query sequence)와 변화의 추세가 유사한 시퀀스들을 검색하는 연산이다[Agr93][Fal94][Raf97]. 유사 검색은 시계열 데이터베이스를 기반으로 하는 데이터 마이닝(data mining) 및 데이터 웨어하우징(data warehousing) 분야에서 중요한 연산으로 사용된다[Che96][Raf97][Loh00][Kim01].

유사 검색은 크게 전체 매칭(whole matching)과 서브시퀀스 매칭(subsequence matching)으로 분류된다[Agr93][Fal94][Par01]. 전체 매칭은 모든 데이터 시퀀스들과 질의 시퀀스의 길이가 항상 동일하다는 전제하에 질의 시퀀스와 유사한 시퀀스를 데이터베이스로부터 검색한다. 반면, 부분 매칭은 다양한 길이의 데이터 시퀀스들이 존재하는 것을 허용하며, 데이터베이스로부터 질의 시퀀스와 유사한 서브시퀀스를 포함하는 시퀀스와 그 시퀀스 내에서의 유사 서브시퀀스의 오프셋을 검색한다. 이와 같이, 서브시퀀스 매칭은 길이에 대한 제약이 없으므로 다양한 실제 응용 분야에서 널리 사용된다.

본 논문에서는 효과적인 서브시퀀스 매칭을 위한 후처리 과정의 최적화 방안을 제안하고자 한다. 제안된 기법의 기본 전략은 후보 윈도우들의 후처리 작업을 일괄 처리 방식으로 수행함으로써 발생 가능한 모든 중복 작업을 제거하는 것이다. 제안된 기법의 성능 개선 효과를 정량적으로 규명하기 위하여 실험을 통한 기존 기법과의 성능을 비교한다.

본 논문에 구성은 다음과 같다. 제 2장에서는 관련 연구로서 기존의 서브시퀀스 매칭 기법에 대하여 간략히 요약한다. 제 3장에서는 본 논문의 연구 동기로서 기존 기법의 후처리 과정에서 나타나는 공통적인 성능상의 문제점을 지적한다. 제 4장에서는 이러한 문제점을 근본적으로 해결할 수 있는 새로운 기법을 제시한다. 제 5장에서는 제안된 기법의 성능 개선 효과를 보이기 위한 성능 평가 결과를 제시한다. 제 6장에서는 논문을 요약하고, 결론을 내린다.

2. 관련 연구

본 장에서는 본 논문에서 해결하고자 하는 문제를 공식적으로 정의하고, 이에 대한 해결 방안을 제시한 기존의 연구 결과를 요약한다.

2.1. 문제 정의

시계열 데이터베이스 D, 길이 n의 질의 시퀀스 Q = (q[0], q[1], ..., q[n-1]), 그리고, 유사 허용치 ε이 주어질 때, 서브시퀀스 매칭 문제는 다음과 같이 정의된다. D에 저장된 길이 N의 임의의 시퀀스 S = (s[0], s[1], ..., s[N-1]) 내에 존재하는 길이 n의 임의의 서브시퀀스 X = (x[0], x[1], ..., x[n-1])가 다음 조건을 만족하면, X를 Q와 유사하다고 간주하여 <S, S내 X의 오프셋>을 반환한다.

1) 본 논문은 한국학술진흥재단 선도연구자 연구비 지원(과제번호: KRF-2000-041-E00258) 및 포디홈네트(4DHomeNet)를 통한 정보통신부 연구비 지원에 의하여 연구되었습니다.

$d(X,Q) \le \varepsilon$, 여기서 $d(X,Q) = \sqrt{\sum_{i=0}^{n-1} (x[i]-q[i])^2}$ [2]

2.2. FRM [Fal94]

참고 문헌 [Fal94]에서는 서브시퀀스 매칭을 위한 해결 방안을 제안하였다. 본 논문에서는 참고 문헌 [Moo01]의 명칭을 따라 이 기법을 FRM이라 부른다. FRM에서는 미리 고정된 크기 w를 갖는 윈도우(window) 개념을 이용한다.

먼저, 각 시퀀스로부터 모든 가능한 위치에서 시작되는 길이 w의 슬라이딩 윈도우(sliding window)들을 추출하고, 각 윈도우를 이산 퓨리에 변환(discrete Fourier transform: DFT)을 이용하여 저차원 공간상의 점으로 변환한다. 본 논문에서는 이 점을 데이터 윈도우 점(data window point)이라 정의한다. 인덱싱의 대상이 되는 윈도우 점들의 수가 매우 많으므로, FRM에서는 이들을 다수의 점들을 포함하는 최소 포함 사각형(minimum bounding rectangle: MBR)들로 구성한 후, 이 MBR들을 다차원 인덱스(multidimensional index)의 하나인 R*-트리[Bec90]에 저장한다[3].

서브시퀀스 매칭을 위해서는 질의 시퀀스로부터 크기 w의 디스조인트 윈도우(disjoint window)들을 추출하고, 윈도우들을 DFT하여 저차원 공간상의 윈도우 점들로 변환한다. 이를 질의 윈도우 점(query window point)이라 한다. 각 윈도우 점에 대하여 $\varepsilon/p^{1/2}$를 허용치로 갖는 범위 질의를 R*-트리 상에서 수행한다. 여기서, $p = \lfloor$ (질의 시퀀스 길이) / w $\rfloor$ 이다. 이러한 범위 질의의 결과로 얻어진 데이터 윈도우 점들을 조사함으로써 최종 결과에 포함될 가능성이 높은 후보 서브시퀀스(candidate subsequence)들을 파악한다. 그 다음, 이 후보 서브시퀀스들을 디스크로부터 액세스하여 질의 시퀀스와의 유클리드 거리를 실제로 계산함으로써 최종적인 진위를 판단한다.

2.3. Dual-Match [Moo01]

FRM에서는 인덱싱을 위한 저장 공간의 오버헤드를 줄이기 위하여 개별적 윈도우 점들 대신 다수의 윈도우 점들을 포함하는 MBR들을 R*-트리 내에 저장한다. 이러한 MBR 내부에는 죽은 공간(dead space)[Bec90]이 존재하게 되므로, 이로 인하여 후보 서브시퀀스의 착오 채택이 발생되며, 이것은 처리 성능 저하로 직결된다[Moo01]. 참고 문헌 [Moo01]에서는 이러한 문제점을 해결하기 위한 방법으로서 이원성 기반 서브시퀀스 매칭(duality-based subsequence matching: Dual-Match)을 제안하였다.

Dual-Match에서는 데이터 시퀀스로부터 슬라이딩 윈도우를 추출하고 질의 시퀀스로부터 디스조인트 윈도우를 추출하는 FRM과는 반대로 데이터 시퀀스로부터는 디스조인트 윈도우를 추출하고 질의 시퀀스로부터는 슬라이딩 윈도우를 추출하는 방식을 사용한다. 이와 같은 역할 교환을 통하여 Dual-Match에서는 R*-트리에 저장할 윈도우들의 수를 FRM의 약 1/w로 줄일 수 있다. 이 결과, MBR들을 저장하는 FRM과는 달리 Dual-Match에서는 윈도우 점 자체를 R*-트리에 저장하는 것이 가능해 진다. 따라서 MBR 내의 죽은 공간으로 인한 후보 서브시퀀스의 착오 채택이 발생되지 않으므로, 처리 성능이 크게 개선된다. 참고 문헌 [Moo01]의 성능 평가 결과에 의하면, Dual-Match의 검색 성능은 FRM과 비교하여 최대 430배까지 개선되는 것으로 나타났다.

2) 즉, 유클리드 거리(Euclidean distance)가 주어진 허용치 ε 이하인 두 서브시퀀스 X와 Q를 유사하다고 간주한다.

3) 현재, 시계열 데이터베이스에서 가장 널리 사용되는 다차원 인덱스는 R*-트리이다. 따라서 본 논문에서는 다차원 인덱스와 R*-트리라는 용어를 혼용한다.

3. 연구 동기

본 장에서는 FRM 및 Dual-Match에서 수행하는 공통적인 처리 과정을 요약하고, 후처리 과정(post-processing step)에서 발생하는 성능상의 문제점을 지적한다.

3.1. 서브시퀀스 매칭 처리 과정

그림 1은 인덱스 검색과 후처리로 구성되는 서브시퀀스 매칭의 처리 과정을 나타낸 것이다. 큰 이등변 삼각형은 R*-트리를 나타내며, 아래쪽의 사각형은 각 시퀀스를 의미한다.

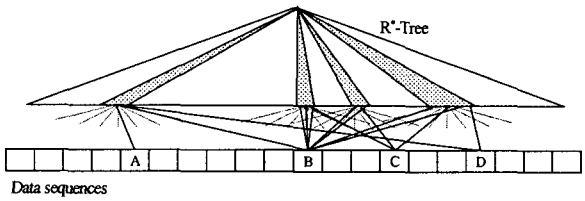

그림 1. 서브시퀀스 매칭의 처리 과정.

인덱스 검색은 각각의 질의 윈도우 점에 대하여 범위가 $\varepsilon/p^{1/2}$인 범위 질의를 R*-트리에 수행하는 과정이다. 그림 1에서 점으로 채워진 네 개의 삼각형은 각 질의 윈도우 점을 이용하여 범위 질의를 처리할 때, R*-트리 내에서 액세스되는 부분을 나타낸다. 각 범위 질의의 결과는 그 범위 질의에서 사용된 질의 윈도우 점과 유클리드 거리가 $\varepsilon/p^{1/2}$ 이내인 데이터 윈도우들의 집합이다. 이들을 후보 윈도우(candidate window)라 정의한다. 이러한 처리를 하는 이론적인 배경은 "서로 다른 두 시퀀스의 유클리드 거리가 ε 이내이기 위한 필요 조건은 대응되는 윈도우 쌍 중 적어도 하나가 $\varepsilon/p^{1/2}$ 이내여야 한다"는 참고 문헌 [Fal94]의 정리에 근거한다.

후처리 과정에서는 먼저 인덱스 검색에서 반환되는 각 후보 윈도우에 대하여 그 윈도우가 속하는 서브시퀀스를 파악한다. 이를 후보 서브시퀀스(candidate subsequence)라 정의한다. 그리고 나서, 이 후보 서브시퀀스가 포함되는 데이터 시퀀스를 디스크로부터 액세스하고, 후보 서브시퀀스와 질의 시퀀스와의 유클리드 거리를 실제로 계산함으로써 착오 채택(false alarm)[Agr93]의 여부를 확인한다.

3.2. 기존 기법들의 후처리 과정의 문제점

그림 1에서 나타난 바와 같이 R*-트리 내에서 윈도우 점들은 모두 독립적으로 관리된다. 또한, 기존의 FRM 및 Dual-Match의 후처리 과정에서는 ***인덱스 검색을 통하여 반환되는 순서***로 각 후보 윈도우가 포함되는 후보 서브시퀀스를 질의 시퀀스와 비교한다. 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 인

덱스 검색의 결과로 반환되는 시점이 다르므로 같은 데이터 시퀀스를 디스크로부터 여러 번 액세스해야 하는 것이다. 그림 1의 예에서 A, B, C, D는 모두 후보 윈도우를 포함하는 후보 시퀀스들이며, 이들은 후처리 과정에서 각각 1, 8, 3, 2회 디스크로부터 액세스되어야 함을 알 수 있다. 이와 같은 디스크로부터의 동일한 시퀀스의 중복 액세스는 전체 서브시퀀스 매칭의 성능을 저하시키는 중요한 원인이 된다.

(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. 그림 2의 경우를 예로 들어보자[4]. 질의 시퀀스 Q내 Qw[1]에서 Qw[k]까지의 k개의 연속적인 윈도우들과 대응되는 데이터 시퀀스 S내 Sw[1]에서 Sw[k]까지의 윈도우들이 모두 후보 윈도우로 반환되면, 질의 시퀀스는 동일한 이 서브시퀀스와 k번 중복하여 비교하게 된다. 이러한 불필요한 중복 비교로 인하여 CPU 시간의 낭비를 초래한다.

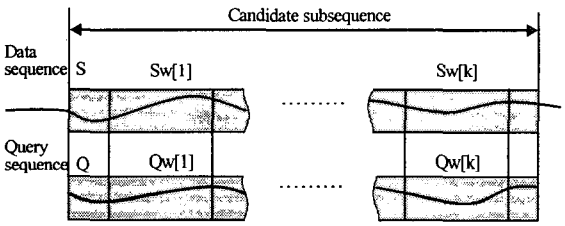

그림 2. 질의 시퀀스와 데이터 서브시퀀스의 비교.

4. 제안하는 기법

본 장에서는 제 3장에서 제시한 문제점들을 해결할 수 있는 새로운 후처리 기법을 제안한다. 제안하는 기법에서 추구하는 궁극적인 목표는 디스크 액세스 및 서브시퀀스 비교 과정에서 발생하는 중복을 제거하는 것이다.

중복의 문제가 발생하는 근본적인 원인은 인덱스 검색의 결과에 포함되는 후보 윈도우들이 무작위 순서대로 반환된다는 데 있다. 본 논문에서는 이러한 문제를 해결하기 위하여 반환되는 후보 윈도우를 위한 후처리 과정을 바로 수행하지 않고, 같은 시퀀스에 속하는 후보 윈도우들, 같은 서브시퀀스에 속하는 후보 윈도우들을 한꺼번에 처리하는 방식을 제안한다.

이를 위한 구체적인 방법은 다음과 같다. 먼저, 후보 윈도우에 대하여 대응되는 <seqID, subseqOffset>를 다중키(multi-attribute key)로 사용하여 이진 트리(binary search tree)에 삽입한다. 여기서 seqID는 이 후보 윈도우가 속하는 데이터 시퀀스의 식별자이다. 또한, subseqOffset은 이 후보 윈도우에 의하여 후보로 추천된 seqID내 서브시퀀스의 시작 오프셋이다. 이 값은 이 후보 윈도우와 매치되는 질의 윈도우의 위치 정보를 이용하여 역으로 계산할 수 있다. 이러한 삽입 연산은 인덱스 검색 과정에서 후보 윈도우가 반환될 때마다 호출된다. 또한, 이러한 삽입 과정에서 삽입하고자 하는 키 값 <seqID, subseqOffset>이 이미 이진 트리 내에 존재하는 경우에는 이 키 값을 삽입하지 않고 무시한다. 예를 들어, 그림 2에서 k개의 후보 윈도우들은 모두 같은 <S, offset>을 키 값으로 가지게 되므로 단 한번만 이진 트리에 삽입된다.

인덱스 검색 과정이 완료되면, 이진 트리 내에는 질의 시퀀스와 비교해야 할 후보 서브시퀀스들이 저장되어 있다. 후처리 과정에서는 이 이진 트리의 중위 순회(in-order traverse)[Knu73]를 통하여 얻어지는 후보 서브시퀀스 순서로 질의 시퀀스와의 비교를 수행한다.

5. 성능 평가

본 장에서는 제안된 기법에 대한 성능 평가 결과를 제시한다. 먼저, 제안된 기법의 최적성을 보이고, 기존 기법과의 비교를 통한 성능 개선 효과를 제시한다.

5.1. 성능 분석

후처리 과정에서 제안된 기법은 이진 트리의 중위 순회를 수행하므로, 이는 <seqID, subseqOffset>의 순서로 후보 시퀀스들을 액세스함을 의미한다. 따라서 같은 시퀀스에 속하는 후보 서브시퀀스들은 연속적으로 비교된다. 따라서 후보 서브시퀀스를 포함하는 시퀀스는 디스크로부터 단 한 번만 액세스된다. 또한, 이진 트리 생성 시 같은 시퀀스에 속하는 중복된 서브시퀀스들은 제거한 바 있다. 따라서 동일한 서브시퀀스의 중복 비교는 발생하지 않는다. 제안된 기법은 반드시 필요한 디스크 액세스와 서브시퀀스 비교를 단 한번만 수행하므로 서브시퀀스 매칭의 후처리 과정을 위한 최적의 방법이다.

5.2. 성능 비교

기존의 기법[5]과의 성능 비교를 위하여 다음과 같은 실험을 수행하였다. 사용된 데이터는 1994년 11월부터 1998년 5월까지 수집한 한국의 주식 데이터 KStock이다. KStock은 길이 1024의 620개 종목의 정규화된 데이터 시퀀스들로 구성된다. 인덱싱을 위하여 사용된 윈도우의 길이는 30, 60, 90의 세 가지이며, 사용된 질의 시퀀스로는 KStock내의 하나의 데이터 시퀀스를 무작위로 선정하여 임의의 위치에서 길이 200만큼을 추출하였다. 또한, 질의를 위한 유사 허용치 ε으로는 21개의 최종 질의 결과를 반환하도록 하는 2.0를 사용하였다. 다차원 인덱스 구성을 위하여 추출된 특징은 각 윈도우를 DFT하여 얻어진 계수 중 큰 에너지를 갖는 앞쪽의 4개를 사용하였다.

표 1은 제안하는 기법과 기존 기법에서 각각 발생하는 시퀀스 액세스 횟수를 (1) 최종 결과로 반환되는 것(true answers), (2) 착오 채택되는 것(false alarms), 그리고 (3) 두 가지 모두를 포함한 것(total) 등 세 가지 측면에서 비교한 것이다. 전체 결과를 나타내는 마지막 두 행에서와 같이, 제안된 기법이 윈도우의 크기에 따라 55배에서 156배까지의 성능 개선 효과를 가지는 것으로 나타났다. 또한, 윈도우가 작을 수록 이러한 성능 개선 효과가 더욱 커지는 것으로 나타났다. 이것은 윈도우가 작아질수록 착오 채택의 수가 많아짐에 따라 후보 시퀀스를 액세스하는 회수가 많아지기 때문이다.

4) 이 그림은 참고 문헌 [Moo01]에서 사용한 것을 인용하였다. 그러나 참고 문헌 [Moo01]에서는 여기서 지적한 문제점을 언급한 것이 아니라 다른 현상을 설명하기 위하여 사용되었던 것임을 밝혀둔다.

5) 기존의 기법 중 나은 성능을 갖는다고 알려진 Dual-Match를 사용하였다.

	방법 \ w	30	60	90
True Answers	ours	21	21	21
	Dual-Match	108	50	21
False Alarms	ours	532	373	250
	Dual-Match	86012	27241	14875
total	ours	553	394	271
	Dual-Match	86210	27291	14896

표 1. 디스크 액세스 회수의 비교.

표 2는 제안하는 기법과 기존 기법에서 각각 발생하는 서브시퀀스 비교 횟수를 (1) 최종 결과로 반환되는 것(true answers), (2) 착오 채택되는 것(false alarms), 그리고 (3) 두 가지 모두를 포함한 것(total) 등 세 가지 측면에서 비교한 것이다. 전체 결과를 나타내는 마지막 두 행에서와 같이, 제안된 기법이 윈도우의 크기에 따라 37%까지의 성능 개선 효과를 가지는 것으로 나타났다. 또한, 이러한 성능 개선 효과는 윈도우가 작을 수록 커지는 것으로 나타났다. 이것은 윈도우가 작아질수록 동일한 서브시퀀스 내에 속하는 후보 윈도우 수가 많아짐에 따라 Dual-Match에서는 동일한 후보 서브시퀀스를 질의 시퀀스와 비교하는 회수가 많아지기 때문이다.

	방법 \ w	30	60	90
True Answers	ours	21	21	21
	Dual-Match	108	50	21
False Alarms	ours	62602	24434	14548
	Dual-Match	86012	27241	14875
total	ours	62623	24455	14569
	Dual-Match	86210	27291	14896

표 2. 서브시퀀스 비교 회수의 비교.

실험 결과, 디스크 액세스 회수 측면에서의 제안된 기법의 성능 개선 효과가 서브시퀀스 비교 측면과 비교하여 훨씬 큰 것으로 나타났다. 디스크 액세스 비용이 전체 처리 시간의 대부분을 차지한다는 것을 고려하면, 이러한 결과는 매우 바람직한 것이다.

6. 결론

본 논문에서는 서브시퀀스 매칭의 후처리 과정에서 발생하는 기존 기법들의 문제점을 지적하고, 이를 해결할 수 있는 최적의 기법을 제안하였다. 제안된 기법은 이진 트리 내에 후보 시퀀스에 대한 정보를 삽입해 둠으로써 같은 시퀀스에 속하는 후보 윈도우들과 같은 서브시퀀스에 속하는 후보 윈도우들을 함께 처리하는 방식을 사용한다. 이 결과, 반드시 필요한 디스크 액세스와 서브시퀀스 비교를 단 한 번씩만 수행한다. 따라서 제안된 기법은 서브시퀀스 매칭의 후처리 과정을 위한 최적의 방법이다. 제안된 기법의 성능 개선 효과를 검증하기 위하여 실제 주식 데이터를 위한 선행 성능 평가를 수행하였다. 실험 결과에 의하면, 제안된 기법은 기존의 기법과 비교하여 55배에서 156배까지의 성능 개선 효과가 있는 것으로 나타났다. 현재, 다양한 환경에서 제안된 기법이 가지는 성능 개선 효과를 정량적으로 파악하기 위하여 다양한 데이터 및 인자들을 대상으로 하는 실험을 수행하고 있다.

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.

[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.

[Che96] M. S. Chen, J. Han, and P. S. Yu, "Data Mining: An Overview from Database Perspective," IEEE Trans. on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 866-883, 1996.

[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. IEEE 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. IEEE Int'l Conf. on Data Engineering, IEEE ICDE, pp. 263-272, 2001.

[Par01] S. H. Park, S. W. Kim, and W. W. Chu, "Segment-Based Approach for Subsequence Searches in Sequence Databases", In Proc. ACM Int'l. Symp. on Applied Computing, ACM SAC, pp. 248-252, 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.