윈도우를 구성하는 방법의 이원성을 이용한 효율적인 시계열 서브시퀀스 매칭

(Efficient Time-Series Subsequence Matching using Duality in Constructing Windows)

문 양 세 † 노 웅 기 †† 황 규 영 †††

(Yang-Sae Moon) (Woong-Kee Loh) (Kyu-Young Whang)

요 약 서브시퀀스 매칭은 질의 시퀀스와 유사한 서브시퀀스를 가지는 데이타 시퀀스와 해당 서브시퀀스의 위치를 찾는 문제이다. 본 논문에서는 윈도우를 구성하는 방법의 이원성을 이용한 새로운 서브시퀀스 매칭 방법인 *Dual-Match*를 제안하고, 이 방법이 서브시퀀스 매칭의 성능을 획기적으로 향상시킬 수 있음을 보인다. Dual-Match는 윈도우를 구성하는 방법에 있어서 Faloutsos 등이 사용한 방법(간단히 *FRM*이라 한다)의 이원적 접근법이다. 즉, FRM에서는 데이타 시퀀스를 슬라이딩 윈도우로 나누고 질의 시퀀스를 디스조인트 윈도우로 나누는 방법을 사용한 반면, Dual-Match에서는 데이타 시퀀스를 디스조인트 윈도우로 나누고 질의 시퀀스를 슬라이딩 윈도우로 나누는 방법을 사용한다. FRM은 색인에 필요한 저장공간을 줄이기 위하여 개별 점 대신 최소 포함 사각형만을 저장함으로 인하여 많은 착오해답(유사하지 않은 후보 서브시퀀스)을 발생시켰다. *Dual-Match*는 FRM과 비슷한 크기의 저장공간에 개별 점을 직접 저장함으로써 이 문제를 해결한다. 실험 결과, Dual-Match는 많은 경우에 있어서 FRM에 비하여 후보 개수를 크게 줄이고 성능을 향상시켰다. 특히, 선택률이 낮은 경우(10^{-4} 이하)에는 후보 개수를 최대 8800배까지 줄이고, 페이지 액세스 횟수를 최대 26.9배까지 줄였으며, 성능을 최대 430배까지 향상시켰다. 또한, 동일한 크기의 색인을 생성하는데 있어서 Dual-Match는 FRM보다 4.10~25.6배 빠르게 색인을 구성하였다. 이는 색인 구성 시에 CPU 오버헤드의 많은 부분을 차지하는 저차원 변환의 횟수를 FRM에 비해 크게 줄이기 때문이다. 이 같은 결과로 볼 때, Dual-Match는 대용량 데이타베이스에 대한 서브시퀀스 매칭의 성능을 크게 향상시킬 수 있는 획기적인 연구 결과라 믿는다.

Abstract Subsequence matching is a solution to a problem of finding the data sequences containing subsequences similar to a given query sequence and of finding the offsets of these subsequences in the original data sequences. In this paper, we propose a new subsequence matching method, which we named Dual-Match. Dual-Match exploits duality in constructing windows and significantly improves overall performance. From the viewpoint of constructing windows, Dual-Match is a kind of dual approach of the subsequence matching method proposed by Faloutsos et al.(FRM in short). FRM divides data sequences into sliding windows, and the query sequence into disjoint windows. Conversely, Dual-Match divides data sequences into disjoint windows and the query sequence into sliding windows. To save storage space for the index, FRM stores minimum bounding rectangles rather than stores individual points representing windows, which causes a lot of false alarms(i.e., candidates that do not qualify). Dual-Match solves this problem by directly storing points without incurring excessive storage overhead. Experimental results show that, given the same amount of storage space, Dual-Match reduces the number of false alarms as well as improves overall

· 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단의 지원을 받았음.

† 학생회원 : 한국과학기술원 전자전산학과 전산학전공
ysmoon@mozart.kaist.ac.kr
†† 비 회 원 : 한국과학기술원 전자전산학과 전산학전공
woong@mozart.kaist.ac.kr
††† 종신회원 : 한국과학기술원 전산학과 교수, 첨단정보기술연구센터
kywhang@cs.kaist.ac.kr

논문접수 : 2000년 1월 31일
심사완료 : 2000년 7월 5일

performance over FRM. In particular, for low selectivities(less than 10-4), Dual-Match drastically reduces the number of candidates—down to as little as $\frac{1}{8800}$ of that for FRM—, reduces the number of page accesses up to 26.9 times, and improves performance by up to 430-fold. Moreover, Dual-Match is 4.10~25.6 times faster than FRM in building indexes of approximately equal sizes. The main reason is that it requires far smaller number of transformations, which are the major part of the CPU overhead, than FRM does. Overall, these results indicate that our approach provides a new paradigm in subsequence matching that improves performance significantly especially in large database applications.

1. 서 론

시계열 데이타(time-series data)는 데이타 마이닝과 데이타 웨어하우스와 같은 데이타베이스의 새로운 응용 분야에서 그 중요성을 더해가고 있다[1]. 시계열 데이타는 각 시간별로 측정한 실수 값의 시퀀스이다. 시계열 데이타의 예로는 주식 데이타, 기업의 성장률, 환율 변동 데이타, 의료 데이타, 날씨 변동 데이타 등이 있다. 시계열 데이타베이스에 저장된 시계열 데이타를 **데이타 시퀀스**(*data sequence*)라 부르며, 사용자에 의해 주어진 **질의 시퀀스**(*query sequence*)와 유사한 데이타 시퀀스를 검색하는 방법을 **유사 시퀀스 매칭**(*similar sequence matching*)이라고 한다[2,3]. 컴퓨터의 계산 및 저장 능력이 발전함에 따라 많은 양의 시계열 데이타를 활용하고자 하는 연구가 활발하게 이루어져 왔으며, 특히 시계열 데이타에 대한 유사 시퀀스 매칭은 데이타 마이닝의 중요한 분야로 자리잡고 있다[1~9].

유사 시퀀스 매칭에 대해서는 여러 가지 유사 모델(similarity model)이 연구되었다. 본 논문에서는 유클리디안 거리에 기반한 모델[2, 3, 5]을 사용한다. 길이 n인 두 시퀀스 X = {X[1],X[2],...,X[n]}와 Y = {Y[1],Y[2],...,Y[n]}의 유클리디안 거리 D(X,Y)는 다음과 같이 정의된다.

$$D(X, Y) = \sqrt{\sum_{i=1}^{n} (X[i] - Y[i])^2} \qquad (1)$$

두 시퀀스 사이의 거리인 D(X,Y)가 사용자가 제시한 **허용치**(*tolerance*)인 ε 이하이면 시퀀스 X와 시퀀스 Y는 **유사**(*similar*)하다고 한다[2]. 본 논문에서는 시퀀스 X와 Y 사이의 거리가 ε 이하이면 X와 Y는 **ε-매치**(ε-*match*)한다고 정의하고, 길이 n인 시퀀스들간의 유클리디안 거리를 계산하는 연산을 **n 차원 거리 계산**(*n-dimensional distance computation*)이라 정의한다.

유사 시퀀스 매칭은 전체 매칭(whole matching)과 서브시퀀스 매칭(subsequence matching)의 두 가지로 구분한다[3].

• **전체 매칭**: 데이타 시퀀스 S_1, S_2, ..., S_N이 있고, 질의 시퀀스 Q와 허용치 ε이 주어졌을 때, Q와 ε-매치하는 모든 데이타 시퀀스를 찾는다. 이때, 모든 S_i와 Q의 길이는 동일하다.

• **서브시퀀스 매칭**: 제각기 다른 길이를 가지는 데이타 시퀀스 S_1, S_2, ..., S_N이 있고, 질의 시퀀스 Q와 허용치 ε이 주어졌을 때, Q와 ε-매치하는 서브시퀀스를 하나 이상 가지는 데이타 시퀀스 S_i와 이들 서브시퀀스의 위치를 찾는다.

서브시퀀스 매칭은 전체 매칭을 일반화한 것이며[3, 5, 6, 9], 본 논문에서는 이러한 서브시퀀스 매칭 문제를 다룬다.

표 1 주요 표기법

기호	정의/의미
N	데이타 시퀀스의 개수
S_i	i번째 데이타 시퀀스($1 \le i \le N$)
Len(S)	시퀀스 S의 길이
Total_Len	모든 데이타 시퀀스 길이의 합 ($= \Sigma Len(S_i)$)
S[k]	시퀀스 S의 k번째 엔트리 ($1 \le k \le Len(S)$)
S[i:j]	시퀀스 S의 i번째 엔트리에서 j번째 엔트리까지로 구성된 서브시퀀스 (만일 i>j이면, 길이 0인 *NULL* 시퀀스를 의미함)
S[i:k]S[k+1:j]	S[i:k]와 S[k+1:j]로 풀어 쓴 표기법
Q	사용자에 의해 주어지는 질의 시퀀스
ω	슬라이딩/디스조인트 윈도우 크기
D(Q,S)	동일한 길이의 Q와 S의 유클리디안 거리
ε	허용치 (사용자가 제시한 최대 허용 거리)
f	색인에 사용되는 특성의 개수
s_i	시퀀스 S의 i번째 디스조인트 윈도우 ($= S[(i-1)*\omega+1:i*\omega]$, $i \ge 1$)

본 논문에서 사용하는 주요 표기와 각 표기에 대한 정의 및 의미는 표 1과 같다. 그리고, 시퀀스 S를 크기 ω인 **슬라이딩 윈도우**(*sliding window*)로 나눈다 함은 $1 \le i \le Len(S) - \omega + 1$인 모든 i에 대해 S[i]를 시작위치로 하는 윈도우들을 구성함을 의미하며, 이렇게 구성된 윈도우들을 슬라이딩 윈도우라 정의한다. 다음으로, 시퀀스 S를 크기 ω인 **디스조인트 윈도우**(*disjoint window*)로 나

눈다 함은 $1 \le i \le \frac{Len(S)-\omega+1}{\omega}$ 인 모든 i에 대해 S[(i-1)*ω+1]를 시작위치로 하는 윈도우들을 구성함을 의미하며, 이렇게 구성된 윈도우들을 디스조인트 윈도우라 정의한다.

서브시퀀스 매칭은 Faloutsos 등[3]에 의하여 좋은 해결책이 제시된 바 있다(본 논문에서는 이 방법을 저자들의 이름 첫글자들을 따서 간략히 *FRM*이라 부른다). FRM에서는 데이타 시퀀스를 슬라이딩 윈도우로 나누고, 질의 시퀀스를 디스조인트 윈도우로 나누는 방법을 사용하였다. 이들은 각 슬라이딩 윈도우를 저차원 공간의 점(point)으로 변환하였는데, 본 논문에서는 이를 간단히 **저차원 변환**(*lower-dimensional transform*)이라 부른다. 그리고, 변환된 점의 개수가 너무 많아 각 점을 개별로 저장할 수 없으므로, 휴리스틱(heuristic)을 사용하여 수백~수천 개의 점을 포함하는 최소 포함 사각형(MBR: minimum bounding rectangle)을 구성한 후, 이들 MBR을 다차원 색인인 R*-트리[10]에 저장하였다. 서브시퀀스 매칭을 위해서는 우선 질의 시퀀스를 디스조인트 윈도우로 나누고, 이들 각 윈도우를 저차원 변환하여 변환한 점과 허용치 ε으로 범위 질의(range query)를 구성하였다. 다음으로, 이들 범위 질의로 색인을 검색하여 변환된 점과 ε 범위에 있는 MBR들을 찾아내고, 찾아낸 MBR이 나타내는 서브시퀀스들을 **후보**(*candidate*)—질의 시퀀스와 유사할(ε-매치할) 가능성이 높은 서브시퀀스—로 삼는다. 마지막으로, 데이타베이스를 액세스하여 이들 후보 중에서 실제로 질의 시퀀스와 ε-매치하는 서브시퀀스들을 찾는다.

FRM은 **착오기각**(*false dismissal*)—질의 시퀀스와 유사한(ε-매치하는) 서브시퀀스이나 착오로 기각되는 서브시퀀스—은 발생하지 않으나, 개별 점 대신 MBR만을 저장함으로 인하여 많은 **착오해답**(*false alarm*)—후보 서브시퀀스이나 실제로는 질의 시퀀스와 유사하지 않은(ε-매치하지 않는) 서브시퀀스—이 발생하고, 이에 따라 성능이 저하된다. 본 논문에서는 윈도우를 구성하는 방법의 이원성(간단히 **이원성**(*duality*)이라 부른다)을 이용하여 착오해답을 크게 줄이고 성능을 향상시키는 서브시퀀스 매칭 방법인 *Dual-Match* (Duality-based subsequence Matching)를 제안한다. Dual-Match는 윈도우를 구성하는 방법에 있어서 FRM의 이원적 접근법이다. 즉, FRM에서는 데이타 시퀀스를 슬라이딩 윈도우로 나누고 질의 시퀀스를 디스조인트 윈도우로 나누는 방법을 사용한 반면, Dual-Match에서는 데이타 시퀀스를 디스조인트 윈도우로 나누고 질의 시퀀스를 슬라이딩 윈도우로 나누는 방법을 사용한다. Dual-Match에서는 데이타 시퀀스를 슬라이딩 윈도우가 아닌 디스조인트 윈도우로 나눔으로써 색인에 저장해야 하는 점의 개수가 FRM의 $1/\omega$로 크게 줄어들고, 따라서 MBR을 구성하지 않고 개별 점을 직접 색인에 저장할 수 있다. 서브시퀀스 매칭을 위해서는 질의 시퀀스를 슬라이딩 윈도우로 나누어 저차원의 점으로 변환한 후, 이들 점과 허용치 ε으로 범위 질의를 구성하였다. 그리고 이들 범위 질의로 색인을 검색하여 후보 서브시퀀스를 구한다. 이와 같이 개별 점을 직접 색인에 저장하고 검색함으로써, Dual-Match는 착오해답을 크게 줄이고 성능을 향상시킬 수 있다. 또한, Dual-Match는 색인 구성에 있어서 FRM에 비하여 약 $1/\omega$의 저차원 변환만을 수행하므로, FRM에 비해 빠른 색인 구성이 가능한 특징을 가진다.

본 논문의 구성은 다음과 같다. 제2절에서는 시계열 데이타의 유사 시퀀스 매칭에 대한 기존 연구를 살펴본다. 제3절에서는 본 연구를 수행하게 된 연구 동기를 설명한다. 제4절에서는 본 논문에서 제안하는 이원성에 의거한 서브시퀀스 매칭을 설명한다. 제5절에서는 제안한 방법의 성능 평가를 위한 실험 환경과 실험 결과를 제시한다. 마지막으로, 제6절에서는 결론을 내린다.

2. 관련 연구

본 절에서는 서브시퀀스 매칭에 대한 관련 연구를 소개한다. 제2.1절에서는 전체 매칭 연구를 설명하고, 제2.2절에서는 대표적인 서브시퀀스 매칭 연구인 Faloutsos 등[3]의 FRM을 설명한다. 본 논문에서 다루는 서브시퀀스 매칭 연구는 유클리디안 거리에 기반한 유사 모델[2, 3, 5]을 사용하고 있다. 현재까지 유클리디안 거리에 기반한 유사 모델하에서 서브시퀀스 매칭을 수행한 연구는 FRM이 유일하다. 그리고, 이 유사 모델 이외의 다양한 유사 모델, 특히 이동 평균 변환, 쉬프팅 및 스케일링, 정규화 변환, 시간 왜곡 변환 등의 전처리 변환을 지원하는 유사 모델에서의 유사 시퀀스 매칭에 대해서는 참고문헌[1, 4, 6, 7, 9, 11 ~13]를 참조한다.

2.1 전체 매칭

시계열 데이타에 대한 유사 시퀀스 매칭은 Agrawal 등[2]에 의하여 처음 제안되었다. 이들은 데이타 시퀀스와 질의 시퀀스의 길이가 동일한 경우인 전체 매칭 문제를 해결하였는데, 그 개략적인 방법은 다음과 같다.

우선, 길이 n인 데이타 시퀀스를 이산 퓨리에 변환(DFT: discrete fourier transform)하여 주파수(frequency) 도메인으로 매핑한 후, f($\le$ n)개의 특성(feature)을 추출하고, 이를 f 차원의 R*-트리[10]에 저장한다. 이렇게 특

성을 추출하는 이유는 다차원 색인의 차원이 증가함에 따라 저장 공간 및 검색 시간 비용이 지수적으로 증가하는 문제[14, 15]로 인하여, 고차원인 시퀀스를 다차원 색인인 R*-트리에 직접 저장하기 어렵기 때문이다. 질의 시퀀스 역시 동일한 방법으로 f 차원 점으로 변환하고, 변환한 점과 허용치 ε을 사용하여 범위 질의를 구성한다. 그리고, 구성한 범위 질의로 R*-트리를 검색하여, ε-매치하는 모든 점들을 찾아 후보집합(candidate set)을 구한다. 이렇게 후보집합을 구하면 착오기각은 발생하지 않지만, 시퀀스 길이 n 대신 f개의 특성만을 사용함으로 인하여 착오해답이 발생할 수 있다. 따라서, R*-트리에 대한 검색 결과로 얻은 각 후보 시퀀스들에 대해서는 실제 데이타 시퀀스를 액세스(access)하고 질의 시퀀스와의 거리를 조사하여 착오해답을 제거하는데, 이 과정을 **후처리 과정**(*post-processing step*)이라 한다[2].

DFT 변환 후 특성을 추출하는 방법과 같이 차원 감소(dimensionality reduction)를 위해 사용하는 함수를 **특성 추출 함수**(*feature extraction function*)라 한다[3]. 그런데, 특성 추출 함수를 유사 시퀀스 매칭에 사용하기 위해서는 보조정리 1과 같이 이를 사용함으로 인한 착오기각이 없어야 한다.

보조정리 1 [3] 범위 질의에 의한 착오기각이 없음을 보장하기 위해서는, 시퀀스 S_i에 대한 특성 추출 함수 F(Si)가 다음 식 (2)를 만족하여야 한다.

$$D_2(F(S_1), F(S_2)) \le D_1(S_1, S_2) \qquad (2)$$

여기에서, D_1(X,Y)와 D2(X,Y)는 두 시퀀스 X와 Y에 대한 거리 함수(distance function)이다. 보조정리 1에서 D_1(X,Y)는 특성 추출 함수를 적용하기 전의 거리 함수이고, D_2(X,Y)는 특성 추출 함수를 적용한 후의 거리 함수이다.

DFT 변환은 D_1(X,Y) 및 D_2(X,Y)가 두 시퀀스 X와 Y에 대한 유클리디안 거리 함수인 경우 보조정리 1을 만족하므로 많은 유사 시퀀스 매칭의 특성 추출 함수로 사용되었다[1~3, 8]. 그리고, 최근의 Chan과 Fu[5]는 Haar Wavelet 변환(간단히 Wavelet 변환이라 한다)을 특성 추출 함수로 사용한 유사 시퀀스 매칭 방법을 제안하였다. 이들은 Wavelet 변환 역시 D_1(X,Y) 및 D_2(X,Y)가 유클리디안 거리 함수인 경우 보조정리 1을 만족함을 보였다.

2.2 서브시퀀스 매칭

Faloutsos 등[3]은 Agrawal 등[2]의 전체 매칭을 일반화한 서브시퀀스 매칭 방법(FRM)을 제안하였다. 이들은 데이타 시퀀스를 슬라이딩 윈도우로 나누고, 질의 시퀀스를 디스조인트 윈도우로 나누는 방법을 사용하였다. 본 절에서는 FRM을 1) 질의 시퀀스 길이가 윈도우 크기와 동일한 경우, 2) 질의 시퀀스가 정확히 p(≥1)개의 디스조인트 윈도우로 구성되는 경우, 그리고 3) 질의 시퀀스를 p개의 디스조인트로 나누었을 때 나머지가 있는 경우의 세 가지로 나누어 설명한다.

첫째, 질의 시퀀스 길이가 윈도우 크기와 동일한 경우를 설명한다. 이 경우는 질의 시퀀스와 ε-매치하는 슬라이딩 윈도우를 찾는 문제가 되므로, Agrawal 등의 전체 매칭 방법을 사용하여 해결할 수 있다. 즉, 특성 추출 함수를 사용하여 각 슬라이딩 윈도우를 f 차원의 점으로 변환한다. 그리고, 질의 시퀀스 역시 f 차원의 점으로 변환하고, 이 점과 허용치 ε으로 범위 질의를 구성한다. 마지막으로, 범위질의를 수행하여 후보집합을 구하고 후처리 과정을 통하여 착오해답을 제거한다. 그런데, 데이타 시퀀스를 슬라이딩 윈도우로 나누므로 대략 Total_Len개의 f 차원 점이 생성되고, 이들 점 모두를 저장하기 위해서는 원래 데이타 시퀀스 저장공간보다 약 f배 많은 저장공간이 필요하다. 또한, 이를 저장하는 R*-트리의 높이가 커져 검색이 느려지고 이로 인하여 순차 검색보다도 성능이 저하된다[3]. 이러한 문제를 해결하기 위하여 FRM에서는 개별 점을 직접 저장하지 않고, 수백~수천 개의 점을 포함하는 MBR을 구성하여 이들 MBR만을 R*-트리에 저장하는 방법을 사용하였다.

MBR을 구성하기 위해서는 색인에 대한 디스크 액세스를 최소화하고자 하는 휴리스틱을 사용하였다. 우선, 데이타 시퀀스 S를 Len(S)-ω+1개의 f 차원 점으로 구성된 하나의 트레일(trail)로 변환한다. 그리고, 허용치 ε으로 추정치(**추정 허용치**(*estimated tolerance*)라 정의하고, ε'으로 나타낸다)인 0.25[1]를 사용하여 MBR의 한점에 대한 한계 비용(marginal cost of a point)을 정의하고, 이 비용이 최소화되도록 트레일을 서브트레일(sub-trail)로 분할하였다[3]. 다음으로, 분할한 각 서브트레일로 MBR을 구성하여 서브트레일의 시작 및 끝 위치, 그리고 시퀀스 식별자와 함께 R*-트리에 저장하는 방법을 사용하였다.

둘째, 질의 시퀀스 Q가 정확히 p개의 디스조인트 윈도우로 구성되는 경우, 즉 Len(Q) = pω인 경우에는 다음 보조정리 2를 사용하여 검색을 수행한다.

보조정리 2 [3] 동일한 길이의 시퀀스 S와 Q를 각각 p개의 디스조인트 윈도우 s_i와 q_i($1 \le i \le p$)로 나누었을 때, 두 시퀀스 S와 Q가 ε-매치하면, 적어도 하나 이상의 (s_i, q_i) 쌍이 $\varepsilon/\sqrt{p}$-매치한다. 즉, 다음 조건식이 성립한다.

1) f 차원 색인에서 각 차원의 주소공간을 [0,1)로 정규화 했을 때, FRM에서 사용한 추정 허용치이다.

$$D(S,Q)\le\varepsilon \Rightarrow \bigvee_{i=1}^{p} D(s_i, q_i)\le\varepsilon/\sqrt{p} \quad (3)$$

FRM에서는 보조정리 2에 따라 질의 시퀀스를 p개의 디스조인트 윈도우로 나누고, 각 윈도우를 f 차원에 변환한 점과 $\varepsilon/\sqrt{p}$으로 범위 질의를 구성한 후, R*-트리를 검색하여 후보집합을 구한다. 이렇게 구한 후보 서브시퀀스들은 식 (3)의 필요조건을 만족하므로 착오기각이 발생하지 않는다.

셋째, 질의 시퀀스를 p개의 디스조인트 윈도우로 나누었을 때 시퀀스의 마지막에 나머지가 있는 경우, 즉 Len(Q) = $p\omega$+k ($1 \le k \le \omega$-1)인 경우에는 다음 보조정리 3을 이용한다.

보조정리 3 [3] 동일한 길이의 시퀀스 S와 Q가 ε-매치하면, (S[i:j], Q[i:j])인 어떠한 서브시퀀스 쌍도 ε-매치한다. 즉, 다음 조건식이 성립한다.

$$D(S,Q)\le\varepsilon \Rightarrow D(S[i:j],Q[i:j])\le\varepsilon \quad (4)$$

보조정리 3에 따라 질의 시퀀스 Q[1:pω+k] 대신에 이의 서브시퀀스인 Q[1:pω]를 사용하여 검색하더라도 착오기각이 발생하지 않는다. 그리고, Q[1:pω]는 정확히 p개의 디스조인트 윈도우로 나누어지므로 보조정리 2를 사용하여 서브시퀀스 매칭을 수행할 수 있다.

지금까지 설명한 FRM을 정리하면 다음과 같다. 우선, 데이타 시퀀스를 슬라이딩 윈도우로 나누어 f 차원의 점으로 변환하고, 여러 개의 점을 포함하는 MBR을 구성하여 R*-트리에 저장한다. 다음으로, 질의 시퀀스를 p(= $\lfloor Len(Q)/\omega \rfloor$)개의 디스조인트 윈도우로 나누어 f차원의 점으로 변환하고, 이 점과 $\varepsilon/\sqrt{p}$으로 범위 질의를 구성한다. 그리고 R*-트리를 검색하여 후보집합을 구성한다. 마지막으로, 각 후보 서브시퀀스에 대해서 데이타 시퀀스를 액세스하고 Len(Q) 차원 거리 계산을 통하여 착오해답을 제거하는 후처리 과정을 수행한다.

3. 연구 동기

본 절에서는 FRM의 이원적 방법을 연구하게 된 동기인 착오해답 발생 원인과 이의 해결 방안에 대해서 설명한다. 유사 시퀀스 매칭에서 착오해답이 많아지면 후처리 과정의 디스크 액세스와 Len(Q) 차원 거리 계산을 위한 CPU 연산을 증가시켜 결국 성능이 크게 저하된다. 따라서, 착오해답을 줄이는 것은 성능 향상의 중요한 요소이다. FRM에서 착오해답이 발생하는 원인은 1) 특성 추출 함수의 사용, 2) 보조정리 2와 3의 사용, 그리고 3) MBR만을 색인에 저장의 세 가지가 있다.

첫번째 원인인 저차원 변환의 사용은 ω 차원의 윈도우를 매핑하여 f 차원만을 사용함으로써 발생하는 착오해답을 말한다. 즉, f 차원에서 두 점 사이의 거리는 ε 이하이나, 실제 두 윈도우 간 거리는 ε보다 큰 경우가 발생한다. 이와 같은 착오해답을 줄이기 위해서는 색인에 사용하는 특성의 개수인 f를 늘이거나, 특성을 보다 잘 추출하는 특성 추출 함수를 사용한다. 최근의 Chan 및 Fu[5]의 Wavelet 변환을 이용하는 연구가 이에 대한 좋은 예이다.

두번째 원인은 길이가 긴 질의 시퀀스에 대해서 보조정리 2와 3을 사용함으로써 발생하는 착오해답을 말한다. 즉, 시퀀스 S와 Q를 p개의 디스조인트 윈도우 s_i와 q_i($1\le i\le p$)로 나누었을 때, (s_i,q_i) 쌍은 $\varepsilon/\sqrt{p}$-매치하나, S와 Q 사이의 거리는 ε보다 큰 경우가 발생한다. 이와 같은 착오해답을 줄이기 위해서는 가능한 큰 윈도우를 사용한다. 예를 들어, 방법 A의 윈도우 크기가 방법 B의 윈도우 크기의 두 배라 하자. 그러면, 보조정리 2 혹은 3에 의하여 방법 A에서 후보이면 방법 B에서도 후보가 되나, 방법 B에서 후보라 하더라도 방법 A에서는 후보가 되지 않을 수 있다. 본 논문에서는 이와 같은 효과를 **윈도우 크기 효과**(*window size effect*)라 정의한다. 그러나, 윈도우 크기는 질의 시퀀스 길이보다는 작거나 같아야 하므로[3], 최대 윈도우 크기는 질의 시퀀스 길이에 의하여 결정된다. 이 점은 제안하는 Dual-Match의 최대 윈도우 크기와 관련하여 제4.4절에서 좀 더 상세히 설명한다.

세번째 원인은 개별 점 대신 MBR만을 저장함으로써 발생하는 착오해답을 말한다. 이는 그림 1을 사용하여 설명한다. 그림에서 P_i($1\le i\le 14$)는 데이타 시퀀스를 나눈 슬라이딩 윈도우가 2 차원 공간(f = 2)에 변환된 점을 나타내고, 이들 14개의 점 P_i가 하나의 MBR을 구성하고 있음을 나타낸다. 그리고, Q_1 및 Q_2는 질의 시퀀스를 나눈 디스조인트 윈도우가 변환된 점을 나타낸다. 그림에서 Q_1 및 Q_2는 MBR과 $\varepsilon/\sqrt{p}$-매치하므로, MBR에 포함된 모든 P_i는 후보집합 구성에 사용된다. 그러나, 실제로 Q_1과 $\varepsilon/\sqrt{p}$-매치하는 P_i는 하나도 없으며, Q_2도 P_8 및 P_9를 제외한 나머지 P_i와는 $\varepsilon/\sqrt{p}$-매치하지 않는다. 따라서, 이들은 많은 착오해답 발생의 원인이 된다. 이와 같은 착오해답을 줄이기 위해서는 MBR을 구성하는 개별 점을 직접 색인에 저장하여야 한다. 그림 1의 예에서 점 P_i를 모두 색인에 저장하면, Q_1의 경우 후보 서브시퀀스가 존재하지 않으며 Q_2의 경우 P_8 및 P_9만 후보 서브시퀀스가 되므로, MBR만을 저장함으로 인해 발생했던 착오해답을 줄일 수 있다. 본 논문에서는 이와 같은 효과를 **점 여과 효과**(*point-filtering effect*)라 정의한다. 그러나, FRM의 경우 제2.2절에서 설명한 바와 같이 모든 점을 색인에 저장할 경우 많은 저장공간이 필요하고 성능이 저하되는

문제가 발생한다. 결국, FRM에서는 세번째 원인에 의한 착오해답을 줄이기가 어렵다.

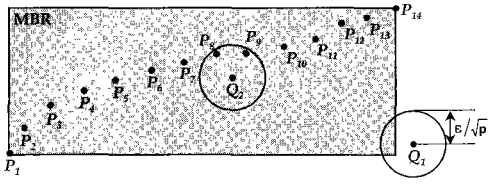


그림 1 MBR 구성으로 인한 착오해답의 발생

지금까지 살펴본 바와 같이 첫번째 및 두번째 원인에 의한 착오해답은 특성 추출 함수와 윈도우 크기에 대한 질의 시퀀스 길이에 의하여 결정되는 사항이다. 그런데, 세번째 원인에 의한 착오해답은 FRM에서 점 여과 효과가 결여되어 발생한 문제이다. 본 논문에서는 점 여과 효과를 활용하여 세번째 원인에 의한 착오해답을 줄이는 새로운 서브시퀀스 매칭 방법인 Dual-Match를 다음 *제4* 절에서 제안한다.

4. 이원성에 의거한 서브시퀀스 매칭

본 절에서는 이원성에 의거한 서브시퀀스 매칭인 Dual-Match를 제안한다. 제4.1절에서는 Dual-Match의 개념과 정리를 설명한다. 다음으로, 제4.2절에서는 색인 구성 알고리즘을 제안하고, 제4.3절에서는 서브시퀀스 매칭 알고리즘을 설명한다. 마지막으로, 제4.4절에서는 최소 질의 시퀀스 길이와 최대 윈도우 크기와의 관계를 논한다.

4.1 개념

제안하는 Dual-Match는 FRM의 이원적 방법으로서, 데이타 시퀀스를 디스조인트 윈도우로 나누고, 질의 시퀀스를 슬라이딩 윈도우로 나누는 접근법을 사용한다. 이렇게 이원적 방법을 사용하는 이유는 색인 구성 및 검색에 개별 점을 직접 사용함으로써 제3절에서 설명한 FRM에서 MBR만을 저장함으로써 발생했던 세번째 원인에 의한 착오해답을 제거하고, 이를 통하여 후처리 과정의 디스크 액세스와 CPU 연산을 크게 줄일 수 있기 때문이다. 제안하는 Dual-Match를 설명하기 위해서 우선 몇 가지 용어를 정의한다. 시퀀스 S의 서브시퀀스 $[i_1:j_1]$과 $S[i_2:j_2]$가 있을 때, $i_1 \geq i_2$이고 $j_1 \leq j_2$이면 $S[i_2:j_2]$는 $S[i_1:j_1]$를 **포함한다**고 정의한다. 그리고, 시퀀스 S를 정해진 디스조인트 윈도우들로 나누었을 때, 이들 중 서브시퀀스 S[i:j]에 포함된 디스조인트 윈도우들을 S[i:j]의 **포함 윈도우**(*included window*)라 정의한다. 동일한 길이의 서브시퀀스들이라 할지라도 그 위치에 따라 포함 윈도우 개수가 다르다. 예를 들어 그림 2에서 서브시퀀스 $S[i_1:j_1]$과 $S[i_2:j_2]$의 길이는 1로 동일하다. 그러나, $S[i_1:j_1]$의 포함 윈도우 개수는 1이고 $S[i_2:j_2]$의 포함 윈도우 개수는 2로, 그 값이 서브시퀀스의 위치에 따라 다름을 알 수 있다. 본 논문에서는 길이 l인 모든 서브시퀀스의 포함 윈도우 개수 중 최소값을 길이 l인 서브시퀀스의 **최소 포함 윈도우 개수**(*minimum number of included windows*)라 정의하고, 이를 p로 나타낸다. 다시 말해서, 길이 l인 서브시퀀스의 최소 포함 윈도우 개수가 p라 함은 길이 l인 모든 서브시퀀스는 그 위치에 관계없이 포함 윈도우 개수가 p 이상임을 의미한다. 본 논문에서는 최소 포함 윈도우 개수 p가 1 이상인 경우만을 고려하며, p는 다음 보조정리 4를 이용하여 구할 수 있다.

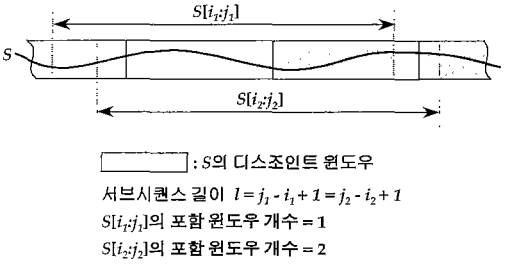


그림 2 길이가 동일한 두 서브시퀀스의 포함 윈도우 개수가 다른 경우

보조정리 4 시퀀스 S를 크기 ω인 디스조인트 윈도우로 나누었을 때, 길이 l인 S의 서브시퀀스의 최소 포함 윈도우 개수 p는 다음과 같다.

$$p = \lfloor (l+1)/\omega \rfloor - 1 \qquad (5)$$

증명: 부록 참조. □

보조정리 4를 사용하면 길이 Len(Q)인 서브시퀀스의 최소 포함 윈도우 개수 p를 $\lfloor (Len(Q)+1)/\omega \rfloor - 1$로 구할 수 있고, 다음 정리 1을 사용하여 유사 서브시퀀스의 후보집합을 구할 수 있다.

정리 1 데이타 시퀀스 S를 크기 ω인 디스조인트 윈도우로 나누고 질의 시퀀스 Q를 같은 크기의 슬리이딩 윈도우로 나누었을 때, 길이 Len(Q)인 S의 서브시퀀스 S[i:j]와 Q가 ε-매치하면, 적어도 하나 이상의 S[i:j]에 포함된 디스조인트 윈도우 $S[i+k:i+k+\omega-1]$ $(0 \leq k \leq Len(Q)-\omega)$와 Q에 포함된 슬라이딩 윈도우 $Q[k:k+\omega-1]$이 $\varepsilon/\sqrt{p}$-매치한다. 여기에서 p는 식 (5)에 의해 구해지는길이 Len(Q)인 서브시퀀스의 최소 포함 윈도우 개수이다.

증명: 증명을 위해 그림 3을 이용한다. 그림에서 질의 시퀀스 Q와 서브시퀀스 S[i:j]가 ε-매치한다고 하자. 최소 포함 윈도우 개수가 p이므로 S[i:j]는 p개 이상의 디스조인트 윈도우를 포함한다. 그림에서 S[i:j]는 p개의 디스조

인트 윈도우 $s_1,...,s_p$를 포함하고, 이들 디스조인트 윈도우의 앞뒤로 s_h(h는 head를 의미함)와 s_t(t는 tail을 의미함)의 서브시퀀스를 포함한다(s_h와 s_t는 NULL 시퀀스일 수 있다). 결국, S[i:j]는 $s_h s_1 \cdots s_p s_t$와 같이 나타낼 수 있다. 동일한 방법으로 Q는 $q_h q_1 \cdots q_p q_t$(Len(q_h) = Len(s_h), Len(q_t) = Len(s_t))와 같이 나타낼 수 있다. 이와 같이 나타내면, 보조정리 2와 3에 의하여 다음 식 (6)이 성립한다.

$$D(S[i:j],Q)\le\varepsilon \Rightarrow D(s_1\cdots s_p, q_1\cdots q_p)\le\varepsilon \quad \text{(보조정리 3에 의함)}$$
$$\Rightarrow \bigvee_{k=1}^{p} D(s_k, q_k)\le\varepsilon/\sqrt{p} \quad \text{(보조정리 2에 의함)} \tag{6}$$

결국, S[i:j]와 Q가 ε-매치하면, S[i:j]는 p(= $\lfloor \frac{Len(Q)+1}{\omega} \rfloor - 1$)개 이상의 디스조인트 윈도우를 포함하고, 이 중 최소한 하나의 윈도우 $s_k(1\le k\le p)$가 Q의 윈도우 q_k와 $\varepsilon/\sqrt{p}$-매치한다. □

질의 시에는 Q의 모든 위치에 대해 슬라이딩 윈도우를 구성하므로, 이들 중에는 정리 1의 윈도우 q_k가 포함되어 있다. 정리 1에 의해서, 데이타 시퀀스를 나눈 디스조인트 윈도우와 질의 시퀀스를 나눈 임의의 슬라이딩 윈도우가 $\varepsilon/\sqrt{p}$-매치할 때, 즉, 식 (6)의 필요조건을만족할 때, 해당 디스조인트 윈도우를 포함하는 서브시퀀스로 후보집합을 구성하면 착오기각 없이 모든 유사 서브시퀀스를 찾을 수 있다.

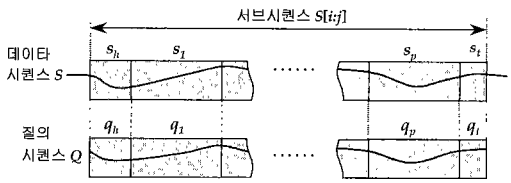


그림 3 윈도우를 이용한 서브시퀀스와 질의 시퀀스의 표현

4.2 색인 구성 알고리즘

본 절에서는 Dual-Match의 색인 구성 알고리즘을 설명한다. 색인 구성 알고리즘은 Build-Index로 그림 4와 같다. Build-Index 알고리즘에서는 데이타 시퀀스로 구성된 데이타베이스를 입력으로 받아, 서브시퀀스 매칭에 사용할 다차원 색인을 구성한다. 알고리즘의 스텝 2.1에서 각 시퀀스 S를 $\lfloor Len(S)/\omega \rfloor$개의 디스조인트 윈도우로 나누는데, 이때 맨 마지막에 남는 길이 ω 미만의 나머지 서브시퀀 $S[\lfloor \frac{Len(S)}{\omega} \rfloor * \omega + 1 : Len(S)]$는 보조정리 3에 의하여 무시할 수 있다. 스텝 2.2.1에서는 각 디스조인트 윈도우 $s_i(1\le i\le \lfloor Len(Q)/\omega \rfloor)$를 f차원 점으로 변환한다. 그리고, 스텝 2.2.2에서는 변환된 점인 *f-point*, 데이타 시퀀스 식별자인 *S-id*, 그리고 디스조인트 윈도우 s_i의 시작 위치인 *dw-offset*(= (i-1)*ω+1)으로 시퀀스를 레코드를 구성한다. 여기에서, 식별자 *S-id*는 색인을 검액세스하기 위해 사용하며, 시작 위치 *dw-offset*은 시퀀스 S에서 해당 서브시퀀스의 위치를 알아내는데 사용한다.

Dual-Match는 디스조인트 윈도우를 매핑한 개별 점을 FRM과 비슷한 크기의 저장공간을 사용하여 직접 색인에 저장할 수 있는 중요한 장점을 가진다. Dual-Match에서는 데이타 시퀀스를 디스조인트 윈도우로 나눔으로써 약 Total_Len/ω개의 점이 생성되고, 따라서, 색인에 필요한 저장공간은 원래 데이타 시퀀스 저장공간의 약 f/ω배가 된다. 이는 FRM에서 개별 점을 직접 저장할 경우에 약 Total_Len개 점과 약 f배의 저장공간과 비교할 때, $1/\omega$에 불과한 수준이다. 그리고, 일반적으로 f는 10 이하이고 ω는 100 이상이므로[3, 5], Dual-Match에서 색인에 필요한 저장공간은 원래 데이타 시퀀스 저장공간의 10 ($\frac{f}{\omega}=\frac{10}{100}$) 이하이고, 저장하는 점의 개수는 전체 데이타 시퀀스 길이의 1%($\frac{1}{\omega}=\frac{1}{100}$)이하에 불과하다.

또한, Dual-Match는 색인으로 점 색인 방법(PAM: point access methods)을 직접 사용할 수 있고, 빠른 색인 구성이 가능한 장점이 있다. 다차원 색인 방법은 점을 저장하는 PAM[16-19]과 점을 포함하여 크기(extent)를 가지는 공간 객체(spatial object)를 저장하는 공간 색인 방법(SAM: spatial access methods)[10, 20, 21]으로 나뉜다[22]. Dual-Match는 점을 저장하므로 PAM을 색인으로 사용할 수 있어, 다양한 특징을 가지는 여러 다차원 색인 방법으로 구현이 가능한 장점이 있다. 그리고, Dual-

Build-Index 알고리즘
입력: 데이타 시퀀스들로 구성된 데이타베이스 *db*
출력: 서브시퀀스 매칭에 사용할 *f* 차원의 다차원 색인 *index*
알고리즘:
1 다차원 색인 *index* 를 초기화 한다.
2 데이타베이스 *db* 에 포함된 각 데이타 시퀀스 *S* (식별자 *S-id*) 에 대해서 다음을 수행한다.
2.1 시퀀스 *S* 를 $\lfloor Len(Q)/\omega \rfloor$개의 디스조인트 윈도우로 나눈다.
2.2 각 디스조인트 윈도우 s_i (시작 위치 *dw-offset* = *(i-1)*ω-1*) 에 대해서 다음을 수행한다.
2.2.1 특성 추출 함수를 사용하여 윈도우 s_i 를 *f* 차원의 점인 *f-point* 로 변환한다.
2.2.2 변환한 점과 윈도우 s_i 에 대한 정보를 사용하여 < *f-point, S-id, dw-offset* > 형태의 레코드를 구성한다.
2.2.3 구성한 레코드를 *f-point* 를 키로 하여 *index* 에 삽입한다.

그림 4 색인 구성 알고리즘

Match는 색인 구성 시 CPU 오버헤드의 많은 부분을 차지하는 저차원 변환을 FRM에 비하여 약 $1/\omega$번 수행하므로, FRM에 비해 빠른 색인 구성이 가능하다.

4.3 서브시퀀스 매칭 알고리즘

본 절에서는 제4.2절에서 구성한 다차원 색인을 바탕으로 유사 서브시퀀스를 찾아내는 서브시퀀스 매칭 알고리즘을 설명한다. 제4.2.1절에서는 기본적인 서브시퀀스 매칭 알고리즘을 제안하고, 제4.2.2절에서는 이를 개선한 알고리즘을 제안한다.

4.3.1 기본 서브시퀀스 매칭 알고리즘

Dual-Match의 기본 서브시퀀스 매칭 알고리즘은 Basic-Dual-Match로 그림 5와 같다. Basic-Dual-Match 알고리즘에서는 시계열 데이타베이스, 다차원 색인, 질의 시퀀스 Q, 그리고 허용치 ε을 입력으로 받아, Q와 ε-매치하는 서브시퀀스를 포함하는 데이타 시퀀스와 해당 서브시퀀스의 시작 위치를 출력한다.

Basic-Dual-Match 알고리즘은 크게 초기화, 색인 검색, 그리고 후처리의 세 단계로 구성된다. 첫째, 초기화 단계에서는 보조정리 4를 사용하여 길이 Len(Q)인 서브시퀀스의 최소 포함 윈도우 개수 p를 $\lfloor (Len(Q)+1)/\omega \rfloor - 1$로 구하고, 질의 시퀀스를 Len(Q)-$\omega$+1개의 슬라이딩 윈도우(Q[i:i+$\omega$-1], 1≤i≤Len(Q)-$\omega$+1)로 나눈다. 둘째, 색인 검색 단계에서는 각 슬라이딩 윈도우 Q[i:i+ω-1]을 사용하여 후보집합을 구한다. 우선, 특성 추출 함수를 사용하여 각 슬라이딩 윈도우를 f 차원의 점으로 변환하고, 변환한 점과 $\varepsilon/\sqrt{p}$으로 범위 질의를 구성한다. 다음으로, 다차원 색인에 범위 질의를 수행하여, 슬라이딩 윈도우 Q[i:i+ω-1]을 매핑한 점과 $\varepsilon/\sqrt{p}$ 범위 내에 있는 점들을 찾아 해당 윈도우의 시작 위치인 i와 함께 후보집합에 포함시킨다. 셋째, 후처리 단계에서는 후보집합의 각 레코드에 대해서, 스텝 3.1에서 후보 서브시퀀스 *sub-S*를 데이타베이스로부터 읽어온다. 다음으로, 스텝 3.2에서는 *sub-S*와 질의 시퀀스 Q와의 Len(Q)-차원 거리 계산을 수행한다. 마지막으로, 스텝 3.3에서는 거리 계산 결과에 따라 ε-매치하는 서브시퀀스만을 취함으로써 착오해답을 제거한다. 그리고, ε-매치하는 각 서브시퀀스 *sub-S*에 대해서, *sub-S*를 포함하는 데이타 시퀀스 S의 식별자 *S-id*와 *sub-S*의 시작 위치를 출력한다.

Basic-Dual-Match 알고리즘은 MBR 대신 개별 점을 사용함으로써, 즉, 점 여과 효과를 이용함으로써 착오해답을 크게 줄일 수 있다. 그러나, Basic-Dual-Match 알고리즘은 각 슬라이딩 윈도우에 대해서 한번씩, 총 Len(Q)-ω+1번의 많은 범위 질의를 수행하여야 하고, 이로 인해 성능 저하될 수 있는 문제점이 있다. 이에 따라, 다음 제4.3.2절에서는 이 문제점을 해결하는 개선된 서브시퀀스 매칭 알고리즘을 제안한다.

Basic-Dual-Match 알고리즘
입력: (1) 데이타 시퀀스로 구성된 데이타베이스 *db*
(2) 알고리즘 BuildIndex로 구성한 *f* 차원 색인 *index*
(3) 질의 시퀀스 *Q*와 허용치 ε
출력: *Q* 와 ε-매치하는 서브시퀀스를 포함하는 데이타 시퀀스의 식별자와 해당 서브시퀀스의 위치
알고리즘:
1 초기화 단계: 다음의 초기화 단계를 수행한다.
1.1 최소 포함 윈도우 개수 p 를 $\lfloor (Len(Q)+1)/\omega \rfloor - 1$로 구한다.
1.2 *Q* 를 *Len(Q)*-ω+1개의 슬라이딩 윈도우 $Q[i:i+\omega-1]$로 나눈다.
2 색인 검색 단계: 각 $Q[i:i+\omega-1]$에 대해서 다음을 수행한다.
2.1 특성 추출 함수를 사용하여 $Q[i:i+\omega-1]$를 *f* 차원의 점으로 변환한다.
2.2 변환한 점과 $\varepsilon/\sqrt{p}$으로 색인을 검색할 범위 질의를 구성한다.
2.3 범위 질의로 *index* 를 검색하고, 검색 결과 찾아낸 레코드들을 윈도우 $Q[i:i+\omega-1]$의 시작위치 *i* 와 함께 후보집합에 포함시킨다.
3 후처리 단계: 후보집합에 포함된 각 레코드 <*f-point, S-id, dw-offset*>에 대해서 다음을 수행한다.
3.1 데이타 시퀀스 *S*의 후보 서브시퀀스 *sub-S* 를 *db* 로 부터 읽는다. 이는 *S-id* 와 *sub-S* 의 시작위치로 '*dw-offset* − *i* + 1'을 사용하여 수행된다. 이때, *S-id*와 *dw-offset*는 레코드에 포함되어 있으며, *i* 는 스텝 2.3에서 레코드와 함께 저장된 윈도우의 시작 위치이다.
3.2 읽어 온 후보 *sub-S*와 *Q* 와의 *Len(Q)*-차원 거리 계산을 수행한다.
3.3 계산한 거리가 ε 보다 작으면, *sub-S*가 유사 서브시퀀스이므로 *S-id* 와 *sub-S* 의 시작위치를 출력한다. 반면에, 거리가 ε 보다 크면, 착오해답이므로 제거한다.

그림 5 기본 서브시퀀스 매칭 알고리즘

4.3.2 개선된 서브시퀀스 매칭 알고리즘

개선된 서브시퀀스 매칭 알고리즘인 Enhanced-Dual-Match 알고리즘은 개별 점에 대해 범위 질의를 수행하지 않고, 여러 점을 포함하는 MBR을 구성하여 범위 질의하는 방법을 사용한다. 이는 FRM에서 데이타 시퀀스를 변환한 점에 대해서 MBR을 구성하는 것과 유사한 개념이다. 그러나, FRM에서는 MBR을 구성하는 점에 대한 정보를 유지할 수 없는 반면에 Dual-Match에서는 이들 정보를 유지할 수 있으며, FRM은 데이타 시퀀스를 위해서 MBR을 사용하는 반면에 Dual-Match는 질의 시퀀스를 위해서 MBR을 사용한다는 점이 다르다. 질의 시퀀스를 나눈 한 슬라이딩 윈도우에 대한 검색 결과는 인접한 다른 슬라이딩 윈도우에 대한 검색 결과와 유사하므로, 인접한 여러 슬라이딩 윈도우들이 변환된 점들을 포함하는 MBR로 검색하여 범위 질의 횟수를 줄인다. 이와 같이 MBR을 구성하여 검색하면 개별 점으로 검색한 경우보다 후보집합이 커지는 경향이 있다. 그러나, MBR을 구성하여 검색함에도 불구하고 Basic-Dual-Match 알고리즘과 동일한 크기의 후보집합을 구할 수 있다. 이는 후처리 과정의 디스크 액세스 및 거리 계산 이전에 색인 검색 결과를 사용한 착오해답의 여과(filtering)가 가능하기 때문

Enhanced-Dual-Match 알고리즘
입력: (1) 데이타 시퀀스로 구성된 데이타베이스 *db*
(2) 알고리즘 **BuildIndex**로 구성한 *f* 차원 색인 *index*
(3) 질의 시퀀스 *Q*와 허용치 ε
출력: *Q* 와 ε-매치하는 서브시퀀스를 포함하는 데이타 시퀀스의 식별자와 해당 서브시퀀스의 위치
알고리즘:
1 초기화 단계: 다음의 초기화 단계를 수행한다.
1.1 최소 포함 윈도우 개수 p 를 $\lfloor (Len(Q)+1)/\omega \rfloor - 1$ 로 구한다.
1.2 *Q* 를 $Len(Q)-\omega+1$개의 슬라이딩 윈도우 $Q[i:i+\omega-1]$로 나누고, 각 윈도우를 *f* 차원의 점으로 변환한다.
1.3 변환한 점들로 범위 질의 구성에 사용할 **MBR**들을 구성한다.
2 색인 검색 단계: 각 **MBR**에 대해서 다음을 수행한다.
2.1 **MBR**과 $\varepsilon/\sqrt{p}$으로 색인을 검색할 범위 질의를 구성한다.
2.2 구성한 범위 질의로 *index* 를 검색하고 색인 수준 여과 (**MBR**의 각 점과 검색 결과 얻은 각 점 사이의 거리를 계산하여 $\varepsilon/\sqrt{p}$-매치하는 점들에 대한 레코드를 해당 슬라이딩 윈도우의 시작 위치 *i* 와 함께 후보집합에 포함시킴)를 수행한다.
3 후처리 단계: 후보집합에 포함된 각 레코드 < *f-point, S-id, dw-offset* >에 대해서 다음을 수행한다.
3.1 데이타 시퀀스 *S*의 후보 서브시퀀스 *sub-S* 를 *db* 로 부터 읽는다. 이는 *S-id* 와 *sub-S*의 시작 위치로 *'dw-offset–i+1'*를 사용하여 수행된다. 여기에서, *S-id*와 *dw-offset*는 레코드에 포함되어 있으며, *i* 는 스텝 2.2에서 레코드와 함께 저장된 윈도우의 시작 위치이다.
3.2 읽어 온 후보 *sub-S*와 *Q* 와의 $Len(Q)$-차원 거리 계산을 수행한다.
3.3 계산한 거리가 ε 보다 작으면, *sub-S*가 유사 서브시퀀스이므로 *S-id* 와 *sub-S* 의 시작 위치를 출력한다. 반면에, 거리가 ε 보다크면, 착오해답이므로 제거한다.

그림 6 개선된 서브시퀀스 매칭 알고리즘

이다. 즉, MBR로 검색한 후에 MBR에 포함된 각 점과 검색 결과 얻은 각 점간의 f 차원 거리 계산을 수행하여, $\varepsilon/\sqrt{p}$-매치하는 점들만을 후보집합에 포함시키는 것인데, 이 같은 여과 과정을 **색인 수준 여과**(*index-level filtering*)라 정의한다. 색인 수준 여과가 가능한 이유는 검색에 사용하는 MBR에 포함된 점들을 관리하기 때문이다. 그림 6은 Enhanced-Dual-Match 알고리즘을 나타내는데, Enhanced-Dual-Match 알고리즘도 Basic-Dual-Match 알고리즘과 같이 초기화, 색인 검색, 그리고 후처리의 세 단계로 이루어진다. 초기화 단계에서는 최소 포함 윈도우 개수 p를 구하고, 질의 시퀀스를 슬라이딩 윈도우로 나누어 f 차원에 변환한 후, 여러 개의 점들을 묶어서 MBR을 구성한다. MBR을 구성하는 방법은 1) 제2.2절의 FRM에서 사용한 휴리스틱 방법, 2) 고정 개수의 점으로 MBR을 구성하는 방법, 그리고 3) 모든 점을 하나의 MBR에 포함시키는 방법 등을 생각할 수 있다. 그러나, MBR 구성 방법에 대한 자세한 논의는 본 논문의 초점이 아니므로 향후 연구로 남겨두기로 한다. 일반적으로, 질의 시퀀스의 길이가 긴 경우는 MBR이 너무 커지지 않도록 여러 개의 MBR을 구성하는 것이 효과적이다. 실제 실험결과 2~8개의 MBR을 사용하면 하나의 MBR을 사용하는 경우보다 성능을 향상시킬 수 있다. 그러나, 본 논문에서는 문제를 단순화하기 위하여 하나의 MBR을 구성하는 방법을 사용한다.

색인 검색 단계에서는 후보집합을 구성한다. 우선, MBR과 $\varepsilon/\sqrt{p}$으로 범위 질의를 구성한다. 그리고, 다차원 색인을 검색하여 MBR과 $\varepsilon/\sqrt{p}$ 범위 내에 있는 점들을 찾아내고, 색인 수준 여과를 통하여 후보집합을 구성한다. 후처리 단계는 Basic-Dual-Match 알고리즘과 동일하다.

4.4 최대 윈도우 크기와 최소 질의 시퀀스 길이와의 관계

본 절에서는 보조정리 5에서 최대 윈도우 크기와 최소 질의 시퀀스 길이와의 관계식을 구하고, 그 의미를 논한다.

보조정리 5 주어진 최소 질의 시퀀스 길이를 Min(Q)라 하면, Dual-Match의 최대 윈도우 크기는 $\lfloor (Min(Q)+1)/2 \rfloor$ 이다.

증명: 정리 1에서, 질의 시퀀스와 ε-매치하는 서브시퀀스는 최소한 하나 이상의 디스조인트 윈도우를 포함하여야 한다. 다시 말해서, 보조정리 4의 최소 포함 윈도우 개수 p가 1 이상이어야 한다. 그러면, 다음 과정에 의하여 질의 시퀀스 길이와 윈도우 크기와의 관계식을 구할 수 있다.[2)]

$$\begin{aligned} p = \lfloor (Len(Q)+1)/\omega \rfloor - 1 \geq 1 &\Leftrightarrow \lfloor (Len(Q)+1)/\omega \rfloor \geq 2 \\ &\Leftrightarrow (Len(Q)+1)/\omega \geq 2 \\ &\Leftrightarrow Len(Q)+1 \geq 2\omega \\ &\Leftrightarrow Len(Q) \geq 2\omega - 1 \end{aligned}$$

그러므로, 주어진 최소 질의 시퀀스 길이가 Min(Q)이면, $Len(Q) \geq 2\omega - 1$에 의하여 $\omega \leq (Min(Q)+1)/2$가 성립하며, 따라서 최대 윈도우 크기는 $\lfloor (Min(Q)+1)/2 \rfloor$ 가 된다. □

최소 질의 시퀀스 길이가 동일하다면, Dual-Match의 최대 윈도우 크기는 FRM의 최대 윈도우 크기의 약 1/2이다. FRM에서 최대 윈도우 크기는 최소 질의 시퀀스 길이 Min(Q)와 동일한 반면에[3], Dual-Match에서의 최대 윈도우 크기는 $\lfloor (Min(Q)+1)/2 \rfloor$ 이기 때문이다. 그런데, 제3절에서 설명한 윈도우 크기 효과에 의하여 윈도우 크기가 작을수록 많은 착오해답이 발생한다. 따라서, 같은 최소 질의 시퀀스 길이에 대하여 Dual-Match는 FRM보다 약간 많은 착오해답을 생성한다. 그러나, Dual-Match에서는 점 여과 효과를 통하여 보다 많은 착오해답을 줄임으로써, 전체적으로 FRM보다 착오해답을 줄이고 성능을 향상시킨다.

2) 관계식을 구하는 과정에서 '2'가 정수이므로 "$\lfloor (Len(Q)+1)/\omega \rfloor \geq 2 \Leftrightarrow (Len(Q)+1)/\omega \geq 2$"가 성립한다.

5. 성능 평가

본 절에서는 제안한 Dual-Match와 FRM의 성능 평가 결과를 설명한다. 제5.1절에서는 성능 평가를 수행한 실험 데이타와 실험 환경을 소개하고, 제5.2절에서는 실험 결과를 설명한다.

5.1 실험 데이타 및 실험 환경

Dual-Match의 우수성을 입증하기 위하여 세 가지 종류의 데이타를 사용하여 많은 실험을 수행하였다. 사용한 데이타는 하나의 긴 데이타 시퀀스로 구성된 것으로서, 이는 여러 개의 데이타 시퀀스로 구성된 경우와 동일한 효과를 가진다. 첫번째 데이타는 FRM[3]에서 사용한 실제 주식 데이타[3]로서 329,112개의 엔트리로 구성되어 있다. 이 데이타를 *STOCK-DATA*라 한다. 두번째 데이타는 데이타 시퀀스의 시작 엔트리를 1.5로 하고, 각 엔트리에 (-0.001, 0.001) 사이의 임의의 값 하나를 더하여 다음 엔트리를 구하는 방식으로 생성된 500만개의 랜덤 워크 데이타(random walk data)이다. 이 데이타 역시 FRM에서 사용한 것으로서 이를 *WALK-DATA*라 한다. 마지막 데이타는 100만개로 구성된 유사 주기를 가지는 시계열 데이타(pseudo periodic synthetic time-series data)[4] 이를 *PERIODIC-DATA*라 한다. PERIODIC-DATA는 유사한 서브시퀀스가 큰 주기를 가지고 반복하여 나타난다. 그리고, STOCK-DATA와 WALK-DATA는 데이타 시퀀스내의 인접한 엔트리들의 변화가 크지 않은 반면에, PERIODIC-DATA의 경우 인접한 엔트리들의 변화가 비교적 큰 특징을 가진다.

실험은 512M 바이트 메모리를 가진 SUN Ultra 60 워크스테이션에서 수행하였다. UNIX 시스템 자체의 버퍼링 효과를 피하고 실제 디스크 I/O를 보장하기 위하여 데이타 및 색인 화일에 대해 원시 디스크(raw disk)를 사용하였으며, 데이타 및 색인 페이지의 크기는 4096 바이트로 하였다. 다차원 색인으로는 FRM과 Dual-Match 모두 R*-트리[10]를 사용하였다. 그리고, 특성 추출 함수로는 DFT 변환과 Wavelet 변환을 사용하였다. 최소 질의 시퀀스 길이는 512로 하여, FRM의 윈도우 크기는 512로 하였으며, Dual-Match의 윈도우 크기는 256으로 하였다. 특성은 FRM과 같은 개수인 6개[5]를 사용하였다. 질의 시퀀스의 길이는 512, 768, 및 1024를 사용하되, 아래의 식 (7)에서 정의될 선택률(selectivity) 각 범위에 골고루 분포되도록 하였다.

FRM에서 MBR을 구성하는 점의 평균 개수는 제2.2절에서 설명한 바와 같이 트레일 분할을 위한 휴리스틱에 사용하는 추정 허용치 ε'에 따라 달라진다. 그리고, MBR을 구성하는 점의 평균 개수에 따라 착오해답의 개수 및 색인 크기가 변한다. 실험에서는 공평한 비교를 위하여 두 방법의 색인 크기와 저장공간의 크기를 비슷하게(차가 10% 미만이 되도록) 하였다. 이는 추정 허용치 ε'을 조절하여 FRM에서 MBR을 구성하는 점의 평균 개수가 Dual-Match의 윈도우 크기와 비슷하게 함으로써, 즉, Dual-Match에서 저장하는 점의 개수와 FRM에서 저장하는 MBR의 개수를 비슷하게 함으로써 가능하다. 이러한 실험 조건에서 Wavelet 변환을 사용한 경우를 사례 A라 하고, DFT 변환을 사용한 경우를 사례 B라 부르기로 한다. 또한, FRM[3]에서 사용한 것과 같이 추정 허용치 ε'이 0.25인 경우에 대해서도 실험하였는데, 이 경우를 사례 C라 부르기로 한다.

실험 결과로는 독립된 컴퓨터에서 두 방법의 후보 개수 비율, 페이지 액세스 횟수[6] 비율, 그리고 수행 시간 비율을 측정하였다. 질의 시퀀스는 데이타 시퀀스의 임의의 위치(random offset)를 시작 엔트리로 한 길이 Len(Q)의 서브시퀀스로 하였으며[3], 노이즈(noise) 효과를 피하기 위하여 같은 길이를 갖는 10개의 다른 질의 시퀀스에 대해서 실험한 후 평균을 취한 값을 실험 결과로 하였다. 질의에 대한 선택률은 모든 가능한 데이타 서브시퀀스 개수에 대한 질의 결과 얻은 유사 서브시퀀스 개수의 비율로서, 다음 식 (7)과 같이 정의된다. 실험에서는 FRM에서 사용한 것과 같은 10-6~10-1 범위의 선택률을 사용하였다[3]. 단, STOCK-DATA는 데이타 시퀀스 길이가 약 33만이므로, 선택률을 약 3.0×10^{-6}부터 실험하였다. 그리고, 원하는 선택률은 각 질의에 대해서 허용치 ε을 조절하여 구하였다.

$$\text{선택률} = \frac{\text{질의 시퀀스 } Q\text{와 } \varepsilon \text{ 이내에 위치하는 서브시퀀스 개수}}{\text{데이타베이스에서 길이 } Len(Q)\text{인 모든 가능한 데이타 서브시퀀스}} \quad (7)$$

5.2 실험 결과

본 절에서는 Dual-Match와 FRM의 성능 평가 결과를 설명한다. 먼저 사례 A의 실험 결과를 자세히 설명하고, 사례 B와 사례 C의 실험 결과를 간략히 소개한다.

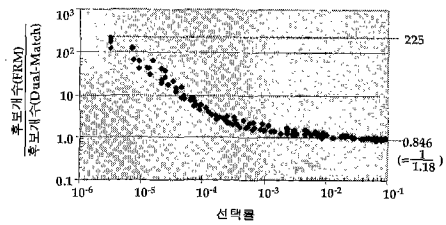
1) STOCK-DATA: 그림 7은 STOCK-DATA에 Wavelet

3) 이 데이타는 ftp://ftp.santafe.edu/pub/Time-Series/data/에서 얻을 수 있다.

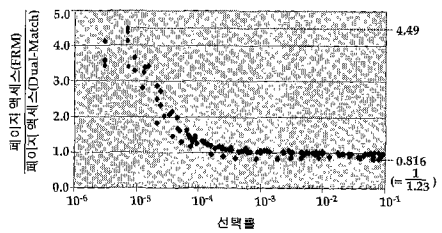
4) 이 데이타는 UC Irvine에서 미국 국립 과학 재단(National Science Foundation)의 지원을 받아 구축 중인 대용량 데이타 집합의 하나로서, http://kdd.ics.uci.edu/databases/synthetic/synthetic.html에서 얻을 수 있다.

5) DFT 변환에서는 첫번째 복소수의 허수부 0대신, 네번째 복소수의 실수부를 사용하였다.

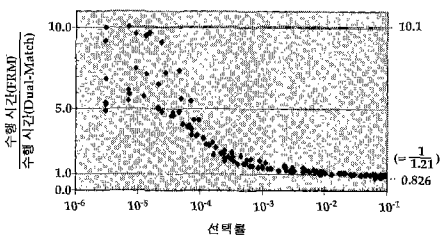
6) 페이지 액세스 횟수 = 데이타 페이지 액세스 횟수 + 색인 페이지 액세스 횟수



(a) 후보개수비율



(b)페이지 액세스 횟수 비율



(c)수행 시간 배율

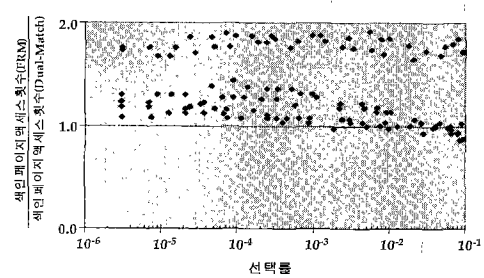
그림 7 STOCK-DATA에 Wavelet 변환을 사용한 경우의 Dual-Match와 FRM의 성능 비교

변환을 사용한 경우의 실험 결과이다. 그림 7의 (a)는 후보 개수 비율을 나타내고, (b)는 페이지 액세스 횟수 비율을 나타내며, (c)는 수행 시간 비율을 나타낸다.

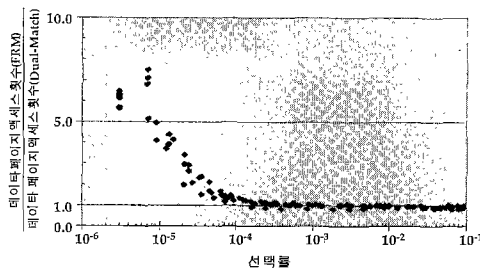
그림에서 보면 선택률이 10^{-3} 이하인 경우에는 Dual-Match가 FRM보다 후보 개수는 최대 225배까지 크게 줄이고, 페이지 액세스 횟수는 최대 4.49배까지 줄였으며, 성능은 최대 10.1배까지 향상시켰음을 알 수 있다. 그리고, 선택률이 10^{-3}~10^{-2}인 경우는 Dual-Match가 세 가지 실험 결과 모두 FRM보다 약간 우수한 것으로 나타났다. 반면에, 선택률이 10^{-2} 이상인 경우는 Dual-Match의 후보 개수가 FRM에 비해 최대 1.18배까지 늘어나고, 페이지 액세스 횟수는 최대 1.23배까지 늘어났으며, 성능은 최대 1.21배까지 약간 나빠진 것으로 나타났다. 선택률이 높은 경우에 후보 개수가 증가하고 성능이 나빠지는 이유는 윈도우 크기 효과에 의하여 Dual-Match의 기본적인 후보집합 크기가 큰 반면, 선택률이 커질수록 후보집합에 대한 착오해답의 비율이 줄어들어 점 여과 효과가 크게 나타나지 않기 때문이다.

그림 7의 실험 결과에서 두 방법의 후보 개수 비율이 페이지 액세스 횟수 및 수행 시간 비율보다 크게 나타났다. 그 이유는 인접한 서브시퀀스들은 유사하고, 이들 인접한 서브시퀀스들은 페이지 크기 만큼씩 한꺼번에 액세스되기 때문이다. 즉, 시퀀스 S의 서브시퀀스 S[i:j]가 질의 시퀀스와 유사하다면, S[i-1:j-1] 및 S[i+1:j+1]을 비롯한 S[i:j]의 인접한 서브시퀀스들은 함께 후보가 되면서 같은 페이지에 저장될 가능성이 크다. FRM의 경우 MBR 구성으로 인하여 개별 점을 사용하는 Dual-Match보다 많은 인접한 서브시퀀스들을 후보집합에 포함하나, 여러 인접한 서브시퀀스들이 동일한 페이지에 저장되고 한꺼번에 액세스되므로 페이지 액세스 및 수행 시간 비율이 후보집합 비율 만큼 크게 나타나지는 않는다.

그림 8의 (a)와 (b)는 두 방법의 색인 페이지 액세스 횟수 비율과 데이타 페이지 액세스 횟수 비율을 각각 나타낸다. 그림 8에서 보는 바와 같이 Dual-Match의 대부분의 페이지 액세스 개선은 데이타 페이지에서 일어나며, 색인 페이지 액세스는 Dual-Match가 FRM보다 약간 좋은 정도이다. 이러한 현상은 다른 모든 실험에서도 유사



(a)색인 페이지 엑세스 횟수 비율



(b)데이타 페이지 엑세스 횟수 비율

그림 8 STOCK-DATA에 Wavelet 변환을 사용한 경우의 Dual-Match와 FRM의 색인 및 데이타 페이지 액세스 횟수 비교

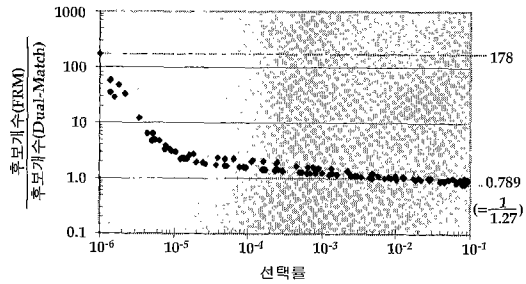
하게 나타났다.

2) WALK-DATA: 그림 9는 WALK-DATA에 Wavelet 변환을 사용한 경우의 실험 결과이다. 그림을 보면 STOCK-DATA의 결과인 그림 7과 유사한 경향을 보임을 알 수 있다. 특히, 선택률이 10^{-4} 이하인 경우는 후보 수를 최대 178배 줄이고, 페이지 액세스 횟수를 최대 5.18배까지 줄였으며, 성능을 최대 14.4배까지 향상시켰다. 단, 선택률이 10^{-2} 이상인 경우에는 Dual-Match의 후보 개수가 FRM에 비해 최대 1.27배(21% 증가)까지 늘어나고, 페이지 액세스 횟수는 최대 1.27배(21% 증가)까지 늘어났으며, 성능은 최대 1.27배(21% 감소)까지 약간 나빠진 것으로 나타났다.

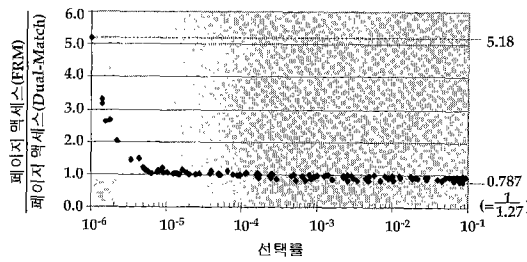
3) PERIODIC-DATA: 그림 10은 PERIODIC-DATA에 Wavelet 변환을 사용한 경우의 실험 결과이다. 이 경우는 두 방법의 세 가지 실험 결과 비율이 그림 7 및 9보다 큰 차이를 나타내고 있다. 그림 10을 보면 선택률이 10^{-4} 이하인 경우에, Dual-Match는 후보 개수를 최대 8800배까지 줄이고, 페이지 액세스 횟수를 26.9배까지 줄였으며, 성능을 최대 430배까지 향상시켰음을 알 수 있다.

PERIODIC-DATA는 비슷한 서브시퀀스가 큰 주기를 가지고 반복하여 나타나지만, STOCK-DATA 및 WALK-DATA에 비하여 인접한 엔트리들의 변화가 큰 특징을 가지고 있다. 따라서, 인접한 슬라이딩 윈도우들이라 할지라도 STOCK-DATA 및 WALK-DATA에 비하여 거리가 먼 경우가 많이 발생한다. 그러므로, MBR을 구성하여 저장하는 FRM에서는 동일한 MBR에 포함된 윈도우들이라 할지라도 윈도우들 사이의 거리가 먼 경우가 많아지고, 이들이 함께 후보집합에 포함됨으로써 많은 착오해답이 발생하게 된다. 반면에, Dual-Match에서는 MBR대신 개별 점을 저장 및 검색에 사용함으로써 이러한 문제가 발생하지 않는다. 이와 같은 이유에 의하여 PERIODIC-DATA는 STOCK-DATA 및 WALK-DATA에 비하여 두 방법의 실험 결과 비율이 큰 차이를 나타내게 되었다. 또한, 이는 WALK-DATA보다 STOCK-DATA에서의 실험 결과가 좀더 많은 차이를 나타내는 이유이기도 하다[7]. 즉, WALK-DATA의 경우 이웃한 엔트리가 (-0.001,0.001) 사이에서 변하므로 평균 변화값이 ±0.0005인데 반하여, STOCK-DATA의 평균 변화값은 ±0.0008로 WALK-DATA보다 크기 때문이다.

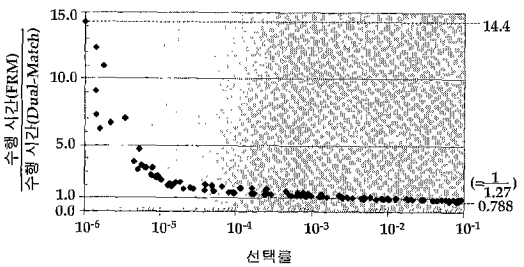
사례 B와 사례 C의 실험 결과도 지금까지 소개한 실험 결과와 유사하게 나타났다. 표 2는 세 가지 경우의 실험 결과를 요약한 것이다. 표 2에서 보듯이, 변환이나 추



(a) 후보 개수 비율



(b) 페이지 액세스 횟수 비율



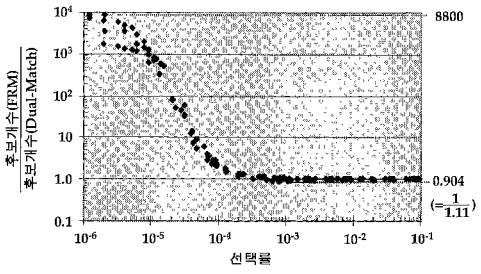
(c) 수행 시간 비율

그림 9 WALK-DATA에 Wavelet 변환을 사용한 경우의 Dual-Match와 FRM의 성능 비교

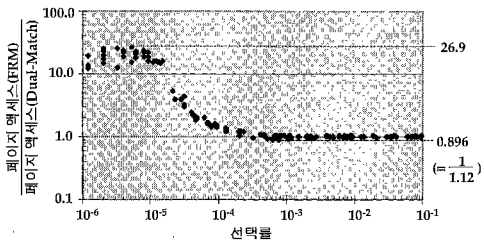
정 허용치를 달리하더라도 선택률이 낮은 경우에는 Dual-Match가 FRM보다 훨씬 우수하게 나타났으며, 선택률이 높은 경우에는 FRM보다 약간 나빠지는 것으로 나타났다.

지금까지의 실험 결과를 종합해 보면, Dual-Match는 선택률이 낮은 경우에 FRM보다 후보 개수를 크게 줄이고 성능을 향상시켰다. 단, 선택률이 높은 경우에는 윈도우 크기 효과로 인하여 후보집합 크기가 약간 커지고, 이에 따라 성능도 약간(29% 미만) 나빠졌다. 일반적으로 데이타 마이닝 분야에서는 대용량 데이타베이스를 주로

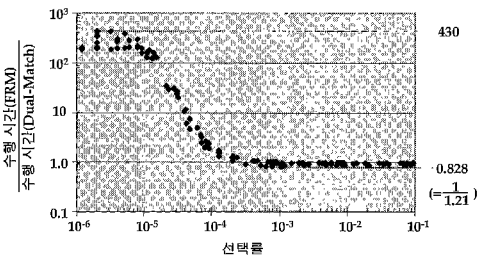
7) 예를 들어, 선택률이 10-5인 경우를 참조한다.



(a) 후보 개수 비율



(b) 페이지 액세스 횟수 비율



(c) 수행 시간 비율

그림 10 PERIODIC-DATA에 Wavelet 변환을 사용한 경우의 Dual-Match와 FRM의 성능 비교

표 2 f 차원 변환과 추정 허용치 ε'을 달리한 경우의 Dual-Match와 FRM의 실험 결과

(FRM/Dual-Match, $10^{-6} \leq$ 선택률 $\leq 10^{-1}$)

실험 방법		STOCK -DATA	WALK -DATA	PERIODIC -DATA
사례 A	후보집합	0.846~225	0.789~178	0.904~8800
	페이지	0.816~4.49	0.787~5.18	0.896~26.9
	수행시간	0.826~10.1	0.788~14.4	0.828~430
사례 B	후보집합	0.859~379	0.798~320	0.886~3710
	페이지	0.824~5.11	0.793~6.98	0.892~9.93
	수행시간	0.964~6.84	0.748~14.5	0.771~41.8
사례 C	후보집합	1.02~876	0.748~39.0	0.868~8790
	페이지	0.885~8.12	0.714~2.76	1.24~20.3
	수행시간	0.965~33.5	0.788~3.99	0.860~255

표 3 Dual-Match와 FRM의 색인 구성 시간 및 저장 공간 비교

(FRM/Dual-Match)

실험방법	색인 구성 시간 비율			평균 저장공간 비율
	STOCK -DATA	WALK -DATA	PERIODIC -DATA	
사례 A	7.60	4.10	4.89	1.03
사례 B	12.0	6.14	6.06	1.10
사례 C	7.16	5.32	25.6	5.19

다룬다. 대용량 데이타베이스를 사용한다면, 길이가 긴 많은 데이타 시퀀스가 시계열 데이타베이스에 존재할 것이고, 일반적으로 사용자는 몇 개의 유사한 서브시퀀스만을 찾을 것이다. 따라서, 실제적인 응용에서는 선택률이 낮은 경우가 많이 쓰일 것이고, 제안하는 Dual-Match가 FRM보다 효과적인 방법이 될 것이다.

색인 구성 시간 및 저장공간에 대한 실험 결과는 표 3과 같다. 표 3에 따르면 Dual-Match는 FRM보다 약 4.10~25.6배 빠르게 색인을 구성함을 알 수 있다. 색인 구성이 빠른 주된 원인은 Dual-Match가 FRM에 비해 약 $1/\omega$의 적은 수의 저차원 변환만을 수행하기 때문이다. 사례 A와 사례 B에 대한 두 방법의 저장공간 오버헤드는 거의 동일하였으며, 사례 C의 경우에는 FRM의 저장공간이 Dual-Match 보다 약 5.19배 더 많은 것으로 나타났다.

6. 결 론

본 논문에서는 시퀀스를 윈도우로 나누는 방법의 이원성을 이용한 새로운 서브시퀀스 매칭 방법인 Dual-Match를 제안하고, Dual-Match의 색인 구성 알고리즘인 Build-Index와 서브시퀀스 매칭 알고리즘인 Basic-Dual-Match 및 Enhanced-Dual-Match를 제안하였다. 그리고, Dual-Match가 기존 Faloutsos 등[3]의 서브시퀀스 매칭 방법(간단히 FRM이라 한다)에 비하여 착오해답을 크게 줄이고 성능을 향상시킴을 보였다. Dual-Match는 데이타 시퀀스를 디스조인트 윈도우로 나누고 질의 시퀀스를 슬라이딩 윈도우로 나누는 방법을 사용하는데, 이는 FRM의 데이타 시퀀스를 슬라이딩 윈도우로 나누고 질의 시퀀스를 디스조인트 윈도우로 나누는 방법의 이원적 접근법이다.

본 논문에서는 FRM에서 착오해답이 발생하는 원인을 분석하고, 그 원인 중의 하나가 FRM의 점 여과 효과의 결여에 있음을 밝혔다. FRM에서는 데이타 시퀀스를 변

환한 모든 점을 직접 색인에 저장할 경우 색인에 필요한 저장공간이 원래 데이타 시퀀스 저장공간의 f 배에 달하여 개별 점을 직접 저장하지 못하고, 여러 점을 포함하는 MBR만을 색인에 저장하였다. 이로 인하여 FRM에서는 점 여과 효과를 활용하지 못하였다. 반면에, Dual-Match에서는 데이타 시퀀스를 디스조인트 윈도우로 나눔으로써 색인에 저장해야 하는 점의 개수가 FRM의 $1/\omega$에 불과하므로, 저장공간의 큰 오버헤드 없이 개별 점을 직접 색인에 저장할 수 있다. 그리고, 개별 점을 직접 저장함으로써 점 여과 효과를 활용할 수 있고, 이 효과에 의하여 Dual-Match는 FRM에 비해 착오해답을 크게 줄일 수 있다.

본 논문에서는 제안하는 Dual-Match가 착오기각 없이 서브시퀀스 매칭을 수행할 수 있음을 정리 1을 통하여 입증하였다. 그리고, 보조정리 5를 통하여 Dual-Match의 최대 윈도우 크기를 유도하였다. 동일한 최소 질의 시퀀스 길이가 주어졌을 때, Dual-Match의 최대 윈도우 크기는 FRM의 최대 윈도우 크기의 약 1/2이 된다. 이와 같이 윈도우 크기가 FRM에 비해 작음으로써, 윈도우 크기 효과에 의하여 선택률이 높은 경우에 Dual-Match의 성능이 FRM보다 약간 저하된다.

본 논문에서는 데이타의 종류, 특성 추출 함수, FRM의 추정 허용치 ε'을 달리하면서 많은 실험을 수행하였다. 실험 결과 Dual-Match는 많은 경우에 있어서 FRM보다 후보집합의 크기를 줄이고 성능을 크게 향상시켰다. 특히, 선택률이 낮은 경우(10^{-4} 이하)에 점 여과 효과에 의하여 후보집합의 크기를 최대 8800배까지 줄이고, 페이지 액세스 횟수를 최대 26.9배까지 줄였으며, 성능을 최대 430배까지 향상시켰다. 선택률이 10^{-4}~10^{-2}인 경우에는 세 가지 실험 결과 모두 Dual-Match가 FRM 보다 약간 더 우수한 것으로 나타났다. 단, 선택률이 높은 경우(10^{-2} 이상)에는 윈도우 크기 효과에 의하여 후보 개수와 페이지 액세스 횟수가 약간 늘고 성능이 약간 저하(29% 미만)되는 것으로 나타났다. 대용량 데이타베이스에서 서브시퀀스 매칭을 수행할 경우, 일반적으로 사용자는 몇 개의 유사 서브시퀀스만을 찾을 것이다. 따라서, 선택률이 낮은 경우에 좋은 결과를 보이는 Dual-Match는 보다 큰 의미를 가질 것이다.

Dual-Match의 또 하나의 중요한 장점은 색인 생성에 있어서 FRM에 비해 성능향상이 탁월하다는 점이다. 실험 결과, 동일한 크기의 색인을 생성하는데 있어서 Dual-Match는 FRM보다 4.10~25.6배 빠르게 색인을 구성하였다. 이는 색인 구성 시에 CPU 오버헤드의 많은 부분을 차지하는 저차원 변환의 횟수를 FRM의 $1/\omega$로 크게 줄일 수 있기 때문이다.

이와 같은 결과를 볼 때, Dual-Match는 대용량 데이타베이스에 대한 서브시퀀스 매칭의 성능을 크게 향상시킬 수 있는 연구 결과로, 서브시퀀스 매칭의 일반적인 해결책으로 여겨졌던 Faloutsos 등의 방법을 대신할 획기적인 연구 결과라 믿는다. Dual-Match는 여러 가지 전처리 변환, 예를 들어, 이동 평균 변환, 쉬프팅 및 스케일링, 정규화 변화 등과 함께 쓰일 수 있을 것으로 예상되며, 이 부분에 대한 자세한 내용는 현재 향후 연구를 수행하고 있다.

참 고 문 헌

[1] Rafiei, D. and Mendelzon, A., "Similarity-Based Queries for Time Series Data," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Tucson, Arizona, pp. 13-25, May 1997.

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

[3] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y., "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.

[4] Agrawal, R., Lin, K.-I., Sawhney, H. S., and Shim, K., "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases," In *Proc. the 21st Int'l Conf. on Very Large Data Bases*, Zurich, Switzerland, pp. 490-501, Sept. 1995.

[5] Chan, K.-P. and Fu, A. W.-C., "Efficient Time Series Matching by Wavelets," In *Proc. the 15th Int'l Conf. on Data Engineering*, Sydney, Australia, pp. 126-133, Feb. 1999.

[6] Chu, K. W. and Wong, M. H., "Fast Time-Series Searching with Scaling and Shifting," In *Proc. the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Philadelphia, Pennsylvania, pp. 237-248, June 1999.

[7] Jagadish, H. V., Mendelzon, A. O., and Milo, T., "Similarity-Based Queries," In *Proc. the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, San Jose, California, pp. 36-45, May 1995.

[8] Rafiei, D., "On Similarity-Based Queries for Time Series Data," In *Proc. the 15th Int'l Conf. on Data Engineering*, Sydney, Australia, pp. 410-417, Feb. 1999.

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

[10] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B., "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.

[11] 노웅기, 김상욱, 황규영, 심규석, "시계열 데이타베이스에서 임의 계수의 이동평균 변환을 지원하는 서브시퀀스 매칭 알고리즘", *한국정보과학회 가을 학술발표 논문집*, Vol. 26, No. 2, pp. 334-336, 1999년 10월.

[12] 노웅기, 김상욱, 황규영, "시계열 데이타베이스에서 인덱스 보간법을 기반으로 정규화 변환을 지원하는 서브시퀀스 매칭", *한국정보과학회 봄 학술발표논문집*, Vol. 27, No. 1, pp. 152-154, 2000년 4월.

[13] Park, S., Chu, W. W., Yoon, J., and Hsu, C., "Efficient Searches for Similar Subsequences of Different Lengths in Sequence Databases," In *Proc. the 16th Int'l Conf. on Data Engineering*, San Diego, California, pp. 23-32, March 2000.

[14] Berchtold, S., Bohm, C., and Kriegel, H.-P., "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.

[15] Weber, R., Schek, H.-J., and Blott, S., "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," In *Proc. the 24th Int'l Conf. on Very Large Data Bases*, New York City, New York, pp. 194-205, Aug. 1998.

[16] Robinson, T. J., "The K-D-B Tree: A Search Structure for Large Multidimensional Dynamic Indexes," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Ann Arbor, Michigan, pp. 10-18, Apr. 1981.

[17] Seeger, B. and Kriegel, H.-P., "The Buddy-Tree: An Efficient and Robust Access Method for Spatial Data Base Systems," In *Proc. the 16th Int'l Conf. on Very Large Data Bases*, Brisbane, Queensland, Australia, pp. 590-601, Aug. 1990.

[18] Whang, K.-Y. and Krishnamurthy, R., Multilevel Grid Files, IBM Research Report RC11516, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, Nov. 1985.

[19] Whang, K.-Y., Kim, S.-W., and Wiederhold, G., "Dynamic Maintenance of Data Distribution for Selectivity Estimation," *The VLDB Journal*, Vol. 3, No. 1, pp. 29-51, Jan. 1994.

[20] Guttman, A., "R-trees: A Dynamic Index Structure for Spatial Searching," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Boston, Massachusetts, pp. 47-57, June 1984.

[21] Sellis, T., Roussopoulos, N., and Faloutsos, C., "The R+-tree: A Dynamic Index for Multidimensional Objects," In *Proc. the 15th Int'l Conf. on Very Large Data Bases*, Brighton, England, pp. 507-518, Sept. 1987.

[22] Gaede, V. and Guenther, O., "Multidimensional Access Methods," *ACM Computing Surveys*, Vol. 30, No. 2, pp. 170-231, June 1998.

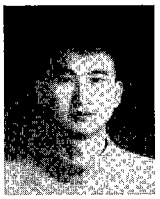
부 록

보조정리 4의 증명: 본 논문에서는 최소 포함 윈도우 개수가 1 이상인 경우만을 고려하므로, 모든 서브시퀀스는 최소한 한 개 이상의 디스조인트 윈도우를 포함한다. 시퀀스 S의 길이 l인 서브시퀀스를 S[i:j](l=j-i+1)라 하고, S[i:j]에 포함된 첫번째 디스조인트 윈도우를 sk(= S[(k-1)*ω+1:k*ω])라 하자. 그러면, S[i:j]는 sk의 첫번째 엔트리인 S[(k-1)*ω+1]을 기준으로 다음과 같이 두 개의 서브시퀀스로 나타낼 수 있다.

$$S[i:j] = S[i:(k-1)*\omega] \; S[(k-1)*\omega+1:j] \quad (8)$$

S[i:j]에 포함된 첫번째 윈도우 sk가 S[(k-1)*ω+1:j]에도 포함되므로, S[i:j]에 포함된 모든 디스조인트 윈도우는 S[(k-1)*ω+1:j]에도 포함된다. 따라서, S[i:j]의 포함 윈도우 개수는 S[(k-1)*ω+1:j]의 포함 윈도우 개수와 같으며, 그 값은 $\lfloor (j-(k-1)*\omega)/\omega \rfloor$ 이다. 그리고, sk가 S[i:j]의 첫번째 윈도우이므로 $(k-1)*\omega \le i+\omega-2$이 성립한다. 그 이유는 만일 $(k-1)*\omega > i+\omega-2$라면 $(k-2)*\omega+1 > i-1$이므로, sk의 바로 이전 디스조인트 윈도우인 sk-1(= S[(k-2)*ω+1:(k-1)*ω])도 S[i:j]에 포함되어 sk가 S[i:j]의 첫번째 윈도우가 되지 못하기 때문이다. 또한, $(k-1)*\omega$ 값이 커질수록 S[i:j]의 포함 윈도우 개수인 $\lfloor (j-(k-1)*\omega)/\omega \rfloor$는 작아지므로, $(k-1)*\omega$가 최대인 $i+\omega-2$일 때 $\lfloor (j-(k-1)*\omega)/\omega \rfloor$는 최소가 된다. 따라서, 길이 l인 서브시퀀스의 최소 포함 윈도우 개수는 다음과 같이 구할 수 있다.

$$\begin{aligned} p &= \min \lfloor (j-(k-1)*\omega)/\omega \rfloor \\ &= \lfloor (j-(i+\omega-2))/\omega \rfloor \quad ((k-1)*\omega = i+\omega-2 \text{ 일 때 최소임}) \\ &= \lfloor ((j-i+1)-\omega+1)/\omega \rfloor \\ &= \lfloor (l-\omega+1)/\omega \rfloor \quad (l = j - i + 1) \\ &= \lfloor ((l+1)-\omega)/\omega \rfloor \\ &= \lfloor (l+1)/\omega \rfloor - 1 \end{aligned}$$



문 양 세

1991년 2월 한국과학기술원 과학기술대 전산학과 학사. 1993년 2월 한국과학기술원 전산학과 석사. 1997년 3월 ~ 현재 한국과학기술원 전자전산학과 전산학 전공 박사과정. 1991년 3월 ~ 현재 현대전자산업(주) 통신사업본부 통신연구소. 관심분야는 데이타 마이닝, Knowledge Discovery, 저장 시스템, Access Method.

노 웅 기

1991년 2월 한국과학기술원 과학기술대 전산학과 학사. 1993년 2월 한국과학기술원 전산학과 석사. 1993년 3월 ~ 현재 한국과학기술원 전자전산학과 전산학 전공 박사과정. 1995년 1월 ~ 2월 일본 NHK 방송기술연구소(동경 소재) 방문(일한재단 초청). 1997년 7월 ~ 8월 미국 Stanford대학 전산학과 방문(Gio Wieder hold 교수 초청). 관심분야는 데이타 마이닝, 정보 검색, 멀티미디어 데이타베이스, 멀티미디어 내용기반 검색.

황 규 영

1973년 서울대학교 전자공학과 졸업(B.S). 1975년 한국과학기술원 전기 및 전자학과 졸업(M.S.). 1982년 Stanford University(M.S.). 1983년 Stanford University(Ph.D.). 1975년 ~ 1978년 국방과학연구소(ADD), 선임연구원. 1983년 ~ 1990년 IBM T.J. Watson Research Center, Research Staff Member. 1992년 ~ 1994년 한국정보과학회 데이타베이스 연구회(SIGDB) 운영위원장. 1995년 한국정보과학회 이사 겸 논문지 편집위원장. 1999년 ~ 2000년 한국정보과학회 부회장. Editior: the VLDB Journal, 1990년 ~ 현재 Editior: Distributed and Parallel Databases: An International Journal, 1991년 ~ 1995년 Editor: International Journal of Geographical Information Systems, 1994년 ~ 현재 Associate Editor: IEEE Data Engineering Bulletin, 1990년 ~ 1993년. 1998년 ~ 2004년 Trustee, The VLDB Endowment. 1999년 ~ 2005년 Steering Committee Member, DASFAA. 1999년 ~ 현재 한국과학기술원 전자전산학과 전산학전공 교수. 1999년 ~ 현재 첨단정보기술연구센터(과학재단 우수연구센터) 소장. 관심분야는 데이타베이스 시스템, 멀티미디어, GIS.