시계열 데이타베이스에서의 서브시퀀스 매칭을 위한 윈도우 구성의 일반화

(Generalization of Window Construction for Subsequence Matching in Time-Series Databases)

문 양 세 † 한 욱 신 †† 황 규 영 †††
(Yang-Sae Moon) (Wook-Shin Han) (Kyu-Young Whang)

요 약 본 논문에서는 서브시퀀스 매칭에서 윈도우 구성의 일반화 개념을 제안하고, 이에 기반한 새로운 서브시퀀스 매칭 방법인 *GeneralMatch*를 제안한다. 기존 연구인 Faloutsos 등의 방법 (간단히 *FRM*이라 한다)은 점 여과 효과의 결여로 인해 많은 착오해답을 발생시켰다. 본 저자들의 *DualMatch*는 점 여과 효과를 발휘하여 성능을 크게 향상시켰으나, 주어진 최소 질의 시퀀스 길이에 대해 최대 윈도우 크기가 작은(FRM의 1/2) 문제가 있었다. GeneralMatch는 DualMatch를 더욱 개선한 방법으로서, 두 방법의 장점을 모두 취한다. 즉, FRM과 같이 큰 윈도우를 사용할 수 있으며, 동시에 DualMatch와 같이 점 여과 효과를 발휘할 수 있다. GeneralMatch는 데이타 시퀀스를 **J-슬라이딩 윈도우**(일반화된 슬라이딩 윈도우)로 나누고, 질의 시퀀스를 **J-디스조인트 윈도우**(일반화된 디스조인트 윈도우)로 나누는 방법을 사용한다. 본 논문에서는 GeneralMatch의 정확성, 즉, GeneralMatch가 착오기각이 발생하지 않음을 증명한다. 또한, 주어진 최소 질의 시퀀스 길이에 대해 GeneralMatch가 바르게 동작하기 위한 최대 윈도우 크기가 있음을 증명한다. 그리고, 페이지 액세스 횟수를 최소로 하는 J 값의 결정 방법을 제안한다. 실제 주식 데이타에 대한 실험 결과, GeneralMatch는 낮은 선택률 범위(10^{-6}~10^{-4})에서 DualMatch에 비해 평균 114%, FRM에 비해 평균 998% 성능을 향상시켰으며, 높은 선택률 범위(10^{-3}~10^{-1})에서도 DualMatch에 비해 평균 46%, FRM에 비해 평균 65% 성능을 향상시켰다.

Abstract In this paper, we present the concept of generalization in constructing windows for subsequence matching and propose a new subsequence matching method, *GeneralMatch*, based on the generalization. The earlier work of Faloutsos et al.(*FRM* in short) causes a lot of false alarms due to lack of the point-filtering effect. *DualMatch*, which has been proposed by the authors, improves performance significantly over FRM by exploiting the point filtering effect, but it has the problem of having a smaller maximum window size (half that of FRM) given the minimum query length. GeneralMatch, an improvement of DualMatch, offers advantages of both methods: it can use large windows like FRM and, at the same time, can exploit the point-filtering effect like DualMatch. GeneralMatch divides data sequences into *J-sliding windows* (generalized sliding windows) and the query sequence into *J-disjoint windows* (generalized disjoint windows). We formally prove that our GeneralMatch is correct, i.e., it incurs no false dismissal. We also prove that, given the minimum query length, there is a maximum bound of the window size to guarantee correctness of GeneralMatch. We then propose a method of determining the value of J that minimizes the number of page accesses. Experimental results for real stock data show that, for low selectivities(10^{-6}~10^{-4}), GeneralMatch improves performance by 114% over DualMatch and by 998% over FRM on the average; for high selectivities(10^{-3}~10^{-1}), by 46% over DualMatch and by 65% over FRM on the average.

* 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단의 지원을 받았음.
† 학생회원 : 한국과학기술원 전자전산학과
ysmoon@mozart.kaist.ac.kr
†† 비 회 원 : 한국과학기술원 전자전산학과
wshan@mozart.kaist.ac.kr
††† 종신회원 : 한국과학기술원 전자전산학과 교수
첨단정보기술연구센터 소장
kywhang@cs.kaist.ac.kr
논문접수 : 2000년 11월 8일
심사완료 : 2001년 4월 24일

1. 서 론

시계열 데이타(time-series data)는 각 시간별로 측정한 실수 값의 시퀀스로, 그 예로는 주식 데이타, 환율 데이타, 날씨 변동 데이타 등이 있다. 시계열 데이타베이스에 저장된 시계열 데이타를 **데이타 시퀀스**(*data sequence*)라 부른다. 그리고, 사용자에 의해 주어진 **질의 시퀀스**(*query sequence*)와 유사한 데이타 시퀀스를 검색하는 방법을 **유사 시퀀스 매칭**(*similar sequence matching*)이라 한다[1,2]. 유사 시퀀스 매칭에 대해서는 여러 가지 유사 모델이 연구되었다. 본 논문에서는 유클리디안 거리에 기반한 모델[1~4]을 사용한다. 이 모델에서는 길이 n인 두 시퀀스 X = {X[1],...,X[n]}와 Y = {Y[1],...,Y[n]}의 유클리디안 거리 $D(X,Y)(=\sqrt{\sum_{i=1}^{n}(X[i]-Y[i])^2})$가 사용자가 제시한 **허용치**(*tolerance*)인 ε 이하이면 X와 Y는 **유사**(*similar*)하다고 한다[1]. 본 논문에서는 D(X,Y)가 ε 이하이면 X와 Y는 **ε-매치**(ε-*match*)한다고 정의한다. 그리고, 길이 n인 시퀀스들간의 유클리디안 거리를 계산하는 연산을 **n-차원 거리 계산**이라 정의한다.

유사 시퀀스 매칭은 전체 매칭(whole matching)과 서브시퀀스 매칭(subsequence matching)의 두 가지로 구분한다[2].

- **전체 매칭**[1] : 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**[2,4] : 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.

표 1 주요 표기법

기호	정의/의미
Len(S)	시퀀스 S의 길이
Total_Len	모든 데이타 시퀀스 길이의 합
S[k]	시퀀스 S의 k번째 엔트리(entry)($1 \le k \le Len(S)$)
S[i:j]	시퀀스 S의 i번째 엔트리에서 j번째 엔트리까지로 구성된 서브시퀀스(만일 i>j이면, 길이 0인 NULL시퀀스를 의미함)
S[i:k]S[k+1:j]	S[i:j]를 S[i:k]와 S[k+1:j]로 풀어 쓴 표기법
s_i	시퀀스 S의 i번째 디스조인트 윈도우 ($= S[(i-1)*\omega+1:i*\omega]$, $i \ge 1$)

서브시퀀스 매칭은 전체 매칭을 일반화한 것이며[2,3,5,6], 본 논문에서는 이러한 서브시퀀스 매칭 문제를 다룬다. 그리고, 본 논문에서 사용하는 주요 표기와 이에 대한 정의 및 의미는 표 1과 같다. 다음으로, 시퀀스 S의 크기 ω인 **슬라이딩 윈도우**(*sliding window*)란 시작 엔트리가 S[i]($1 \le i \le Len(S)-\omega+1$)인 길이 ω의 서브시퀀스이며, 크기 ω인 **디스조인트 윈도우**(*disjoint window*)란 시작 엔트리가 S[(i-1)*ω+1]($1 \le i \le \frac{Len(S)-\omega+1}{\omega}$)인 길이 ω의 서브시퀀스이다.

기존 연구들[2,4]에서의 서브시퀀스 매칭은 다음과 같이 처리된다. 우선, 데이타 시퀀스를 크기 ω의 윈도우로 나누어 f-차원($f \le \omega$) 공간의 점으로 변환한 후(간단히 **저차원 변환**이라 한다), 이들 점을 다차원 색인에 저장한다. 다음으로, 질의 시퀀스를 크기 ω의 윈도우로 나누어 f-차원의 점으로 변환한다. 그리고, 변환한 점과 허용치 ε으로 범위 질의를 구성한 후, 다차원 색인을 검색하여 **후보**(*candidate*, 질의 시퀀스와 ε-매치할 가능성이 높은 서브시퀀스)들을 구한다. 마지막으로, 데이타베이스를 액세스하여 후보들 중에서 실제로 질의 시퀀스와 ε-매치하는 서브시퀀스들을 찾는다.

Faloutsos 등[2]은 데이타 시퀀스를 슬라이딩 윈도우로 나누고 질의 시퀀스를 디스조인트 윈도우로 나누는 서브시퀀스 매칭 방법을 제안하였다(이 방법을 저자들의 이름 첫글자들을 따서 간략히 *FRM*이라 한다). FRM에서는 데이타 시퀀스를 슬라이딩 윈도우로 나누므로 개별점을 직접 저장하기 위한 저장공간의 오버헤드가 매우 커진다. 따라서, 휴리스틱을 사용하여 수백~수천 개의 점을 포함하는 최소 포함 사각형(MBR: minimum bounding rectangle)을 구성하고, 이들 MBR을 다차원 색인에 저장한다. 이러한 FRM은 개별 점 대신 MBR만을 저장함으로 인하여 점 여과 효과가 결여되고, 이에 따라 많은 **착오해답**(*false alarm*, 후보이나 실제로는 질의 시퀀스와 ε-매치하지 않는 서브시퀀스)이 발생하고 성능이 저하되는 문제점이 있다[4]. 여기에서, **점 여과 효과**(*point filtering effect*)란 개별 점을 직접 색인에 저장하고 검색함으로써, 점대점 비교를 통하여 착오해답을 줄이는 효과이다[4].

FRM의 점 여과 효과 결여의 문제를 해결하기 위해서, 본 저자들은 윈도우 구성의 이원성(duality)을 이용한 서브시퀀스 매칭 방법인 *DualMatch*[4]를 제안하였다. DualMatch는 FRM의 이원적 방법으로, 데이타 시퀀스를 디스조인트 윈도우로 나누고 질의 시퀀스를 슬라이딩 윈도우로 나누는 접근법을 사용한다. DualMatch에서는 데이타 시퀀스를 슬라이딩 윈도우가 아닌 디스조인

트 윈도우로 나눔으로써 색인에 저장해야 하는 점의 개수가 FRM의 $1/\omega$로 크게 감소하므로, MBR을 구성하지 않고 개별 점을 직접 색인에 저장한다. 이와 같이 개별 점을 직접 색인에 저장함으로써, DualMatch는 점 여과 효과를 발휘할 수 있으며, 이를 통하여 착오해답을 크게 줄이고 성능을 향상시킨다. 그런데, 최대 윈도우 크기에 있어서 DualMatch는 FRM의 약 1/2에 불과한 문제점이 있다[4]. 동일한 환경인 경우 윈도우 크기가 작을수록 착오해답이 늘어나는 경향이 있는데, 이에 따라 윈도우 크기가 작은 DualMatch는 선택률[1]이 높은 일부 구간(10^{-2} 이상)에서는 FRM보다 성능이 약간 저하된다.

본 논문에서는 윈도우 구성의 일반화 기법을 제안하고, 이에 기반한 서브시퀀스 매칭 방법인 *GeneralMatch* (*Generalized subsequence Matching*)를 제안한다. GeneralMatch는 DualMatch를 더욱 개선한 방법으로서, DualMatch와 같이 개별 점을 직접 색인에 저장하되, DualMatch보다 윈도우 크기가 큰 여러 가지 서브시퀀스 매칭 방법이 가능하다. 이를 위하여, 우선 슬라이딩 윈도우와 디스조인트 윈도우를 일반화한 **J-슬라이딩 윈도우**(*J-sliding window*)와 **J-디스조인트 윈도우**(*J-disjoint window*)를 정의한다. 그리고, 데이타 시퀀스를 J-슬라이딩 윈도우로 나누고 질의 시퀀스를 J-디스조인트 윈도우로 나누는 일반화된 윈도우 구성 방법으로 서브시퀀스 매칭을 수행할 수 있음을 보인다. GeneralMatch는 J 값을 조정함에 따라 많은 서브시퀀스 매칭 방법이 가능하며, 본 논문에서는 페이지 액세스 횟수를 최소화 하는 방향으로 J 값을 결정하는 방법을 제안한다. 그리고, 실험을 통하여 GeneralMatch가 기존의 FRM은 물론 DualMatch보다 더 우수한 서브시퀀스 매칭임을 보인다.

본 논문의 구성은 다음과 같다. 제2절에서는 서브시퀀스 매칭에 대한 기존 연구를 살펴본다. 제3절에서는 본 논문에서 제안하는 윈도우 구성의 일반화와 GeneralMatch를 설명한다. 제4절에서는 제안한 방법의 성능 평가 결과를 제시하고, 마지막으로, 제5절에서 결론을 내린다.

2. 관련 연구

본 절에서는 Agrawal 등[1]의 전체 매칭 연구와 서브시퀀스 매칭 연구인 Faloutsos 등의 FRM[2]과 본 저자들의 DualMatch[4]를 설명한다. 본 논문에서 다루는 서브시퀀스 매칭 연구는 유클리디안 거리에 기반한 유사 모델[1-3]을 사용한다. 그리고, 이 모델 이외의 다른 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 [5-11]를 참조한다.

전체 매칭

Agrawal 등[1]은 데이타 시퀀스와 질의 시퀀스의 길이가 동일한 경우인 전체 매칭 문제를 해결하였는데, 그 개략적인 방법은 다음과 같다. 우선, 길이 n인 데이타 시퀀스를 저차원 변환하여 f(≤n)개의 특성(feature)을 추출하고, 이를 f-차원의 R^*-트리[12]에 저장한다. 이렇게 특성을 추출하는 이유는 다차원 색인의 고차원 문제(high dimensionality problem)[1,13]로 인하여, 고차원인 시퀀스를 다차원 색인인 R^*-트리에 직접 저장하기 어렵기 때문이다. 이와 같이 저차원 변환을 위해 사용하는 함수를 **특성 추출 함수**(*feature extraction function*)라 한다[2]. 질의 시퀀스 역시 동일한 방법으로 f-차원 점으로 변환하고, 변환한 점과 허용치 ε을 사용하여 범위 질의를 구성한다. 그리고, 범위 질의로 R^*-트리를 검색하여, ε-매치하는 모든 점들을 찾아 후보집합을 구한다. 이렇게 후보집합을 구하면 **착오기각**(*false dismissal*, 후보이나 실제로는 질의 시퀀스와 ε-매치하지 않는 데이타 (서브)시퀀스)은 발생하지 않지만, 시퀀스 길이 n대신 f개의 특성만을 사용함으로 인하여 착오해답이 발생할 수 있다. 따라서, R^*-트리에 대한 검색 결과로 얻은 각 후보 시퀀스들에 대해서는 데이타베이스에 저장된 실제 데이타 시퀀스를 액세스하고 질의 시퀀스와의 거리를 조사하여 착오해답을 제거하는데, 이 과정을 **후처리 과정**(*post-processing step*)이라 한다[1].

서브시퀀스 매칭 -FRM

Faloutsos 등[2]은 서브시퀀스 매칭 방법인 FRM을 제안하였다. 본 논문에서는 FRM을 크게 색인 구성 알고리즘과 서브시퀀스 매칭 알고리즘으로 나누어 설명한다.

FRM의 색인 구성 알고리즘에서는 데이타 시퀀스로 다차원 색인을 구성한다. 우선, 데이타 시퀀스를 슬라이딩 윈도우로 나누고, 특성 추출 함수를 사용하여 각 슬라이딩 윈도우를 f-차원의 점으로 저차원 변환한다. 데이타 시퀀스를 슬라이딩 윈도우로 나누므로 대략 Total_Len개의 많은 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 이

1) 선택률=(질의 시퀀스 Q와 ε-매치하는 서브시퀀스 개수)/(데이타베이스에서 길이 Len(Q)인 모든 가능한 데이타 서브시퀀스 개수)

로 인하여 순차 검색보다도 성능이 저하된다[2]. 이러한 문제를 해결하기 위하여 FRM에서는 여러 개의 점을 포함하는 MBR을 구성한 후, 이들 MBR만을 R^*-트리에 저장하는 방법을 사용한다.

FRM의 서브시퀀스 매칭 알고리즘에서는 다음 두 가지 보조정리에 기반하여 질의 시퀀스 Q를 $p(=\lfloor Len(Q)/\omega \rfloor)$개의 디스조인트 윈도우로 나누어 질의하는 방법을 사용한다.

보조정리 1 [2] 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 조건식 (1)이 성립한다.

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

보조정리 2 [2] 동일한 길이의 시퀀스 S와 Q가 ε-매치하면, (S[i:j],Q[i:j])인 어떠한 서브시퀀스 쌍도 ε-매치한다. 즉, 다음 조건식 (2)가 성립한다.

$$D(S,Q) \le \varepsilon \Rightarrow D(S[i:j],Q[i:j]) \le \varepsilon \quad (2)$$

보조정리 1과 2를 질의 시퀀스 Q와 데이타 시퀀스 S의 서브시퀀스 S[i:j]를 사용하여 다시 표현하면 다음 보조정리 3과 같다.

보조정리 3 [2] 데이타 시퀀스 S를 크기 ω인 슬라이딩 윈도우로 나누고 질의 시퀀스 Q를 같은 크기의 디스조인트 윈도우로 나누었을 때, 길이 Len(Q)인 S의 서브시퀀스 S[i:j]와 Q가 ε-매치하면, Q에 포함된 적어도 하나 이상의 디스조인트 윈도우 $q_k(1 \le k \le p)$와 S[i:j]에 포함된 슬라이딩 윈도우 $S[i+(k-1)*\omega : i+k*\omega-1]$이 $\varepsilon/\sqrt{p}$-매치한다. 여기에서 p는 $\lfloor (Len(Q)/\omega \rfloor$이다.

보조정리 3에 따르면, q_k와 $S[i+(k-1)*\omega : i+k*\omega-1]$가 $\varepsilon/\sqrt{p}$-매치할 때, 서브시퀀스 S[i:j]로 후보집합을 구성하면 착오기각 없이 모든 유사 서브시퀀스를 구할 수 있다. FRM은 질의 시퀀스를 디스조인트 윈도우로 나누어 f-차원의 점으로 변환하고, 이 점과 $\varepsilon/\sqrt{p}$으로 범위 질의를 구성한다. 그리고, R*-트리를 검색하여 $\varepsilon/\sqrt{p}$-매치하는 MBR들을 찾아내고, 이들 MBR이 나타내는 서브시퀀스들로 후보집합을 구한다. 마지막으로, 후처리 과정을 통하여 착오해답을 제거한다.

서브시퀀스 매칭-DualMatch

본 저자들이 제안한 서브시퀀스 매칭 방법인 DualMatch [4]도 색인 구성 알고리즘과 서브시퀀스 매칭 알고리즘으로 구성된다. DualMatch의 색인 구성 알고리즘은 FRM과 마찬가지로 데이타 시퀀스로 다차원 색인을 구성하는 기능을 수행한다. 그러나, DualMatch에서는 FRM에서 MBR만을 색인에 저장한 것과 달리 개별 점을 직접 색인에 저장한다. DualMatch에서는 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 수준으로, 개별 점을 직접 색인에 저장하더라도 FRM에서와 같은 저장공간의 증가와 성능저하의 문제가 발생하지 않는다. 그리고, DualMatch는 점을 저장하기 때문에 다차원 색인으로 공간 색인 방법(spatial access methods)[12,14]뿐 아니라 점 색인 방법(point access methods)[15-17]도 사용할 수 있다는 장점이 있다.

DualMatch의 서브시퀀스 매칭 알고리즘에서는 다음 보조정리 4에 기반하여 착오기각 없이 서브시퀀스 매칭을 수행한다.

보조정리 4 [4] 데이타 시퀀스 S를 크기 ω인 디스조인트 윈도우로 나누고 질의 시퀀스 Q를 같은 크기의 슬라이딩 윈도우로 나누었을 때, 길이 Len(Q)인 S의 서브시퀀스 S[i:j]와 Q가 ε-매치하면, S[i:j]에 포함된 적어도 하나 이상의 디스조인트 윈도우 $S[i+k:i+k+\omega-1]$과 Q에 포함된 슬라이딩 윈도우 $Q[k:k+\omega-1]$이 $\varepsilon/\sqrt{p'}$-매치한다. 여기에서 p'은 $\lfloor (Len(Q)+1)/\omega \rfloor - 1$이다.

보조정리 4에 따르면, $S[i+k:i+k+\omega-1]$과 $Q[k:k+\omega-1]$이 $\varepsilon/\sqrt{p'}$-매치할 때, 서브시퀀스 S[i:j]로 후보집합을 구성하면 착오기각 없이 모든 유사 서브시퀀스를 구할 수 있다. DualMatch는 질의 시퀀스를 슬라이딩 윈도우로 나누어 f-차원 점으로 변환하고, 범위 질의의 횟수를 줄이기 위하여 여러 개의 점을 포함하는 MBR을 구성한다. 다음으로, MBR과 $\varepsilon/\sqrt{p'}$으로 범위 질의를 구성하여 다차원 색인을 검색한 후, 색인 수준 여과를 통하여 개별 점을 직접 질의에 사용한 경우와 동일한 후보집합을 구성한다. 여기에서, **색인 수준 여과** (*index-level filtering*)란 MBR을 구성함으로 인해 발생할 수 있는 착오해답을 디스크 액세스 없이 색인 수준에서 제거하는 과정으로, 색인을 검색한 후에 MBR에 포함된 각 점과 검색 결과 얻은 각 점간의 f-차원 거리 계산을 수행하여 ε-매치하는 점들만으로 후보집합을 구성하는 과정이다[4]. 마지막으로, 후처리 과정을 거쳐 착오해답을 제거한다.

보조정리 1과 2에 따르면, 윈도우 크기가 클수록 착오

해답이 줄어들고 윈도우 크기가 작을수록 착오해답이 커지는 경향이 있는데, 이를 **윈도우 크기 효과**(*window size effect*)라 한다[4]. 예를 들어, 방법 A의 윈도우 크기가 방법 B의 윈도우 크기의 두 배라 하자. 그러면, 보조정리 1과 2에 의하여 방법 A의 후보이면 방법 B에서도 후보가 되나, 방법 B에서 후보라 하더라도 방법 A에서는 후보가 되지 않을 수 있다. 따라서, 착오해답을 줄이기 위해서는 가능한 큰 윈도우를 사용하여야 한다. 그런데, 서론에서 이미 언급한 바와 같이 DualMatch는 최대 윈도우 크기가 FRM의 약 1/2에 불과한 문제점이 있다.

3. 서브시퀀스 매칭에서 윈도우 구성의 일반화

본 절에서는 윈도우 구성의 일반화 개념을 제안하고, 이에 기반한 서브시퀀스 매칭인 GeneralMatch를 설명한다. 제3.1절에서는 일반화 기법의 개념을 설명한다. 제3.2절과 제3.3절에서는 GeneralMatch의 색인 구성 알고리즘과 서브시퀀스 매칭 알고리즘을 각각 설명한다. 다음으로, 제3.4절에서는 GeneralMatch의 최대 윈도우 크기와 색인에 저장하는 점의 개수를 구하고 그 의미를 논한다. 마지막으로, 제3.5절에서는 GeneralMatch의 최적 J 값을 찾는 방법을 제안한다.

3.1 개념

본 절에서는 GeneralMatch에서 데이타 시퀀스와 질의 시퀀스를 윈도우로 나누는 방법을 설명하고, General Match가 이 방법을 통하여 착오기각 없이 서브시퀀스 매칭을 수행함을 보인다.

J-슬라이딩 윈도우

GeneralMatch는 데이타 시퀀스를 다음과 같이 정의되는 J-슬라이딩 윈도우로 나눈다.

정의 1 시퀀스 S의 크기 ω인 J-슬라이딩 윈도우($1 \le J \le \omega$)는 시작 엔트리가 $S[(i-1)*J+1]$($1 \le i \le \frac{Len(Q)-\omega}{J}+1$)인 길이 ω의 서브시퀀스로 정의한다. 그리고, S를 J-슬라이딩 윈도우로 나눈다함은 S의 모든 가능한 위치에 대해 J-슬라이딩 윈도우를 구성함을 의미한다.

직관적으로는, 그림 1과 같이 한번에 J씩 이동하면서 S[1], S[J+1], S[2*J+1],... 각각을 시작 엔트리로 하는 크기 ω의 윈도우를 구성하는 방법이다. 따라서, S의 i 번째 J-슬라이딩 윈도우는 $S[(i-1)*J+1:(i-1)*J+\omega]$이고, 본 논문에서는 이를 s_i^J로 나타낸다.

시퀀스 S를 J-슬라이딩 윈도우로 나누었을 때, 서브시퀀스 S[i:j]에 포함된 첫번째 J-슬라이딩 윈도우가 S의 몇 번째 J-슬라이딩 윈도우인지는 다음 보조정리 5를 이용하여 구할 수 있다.

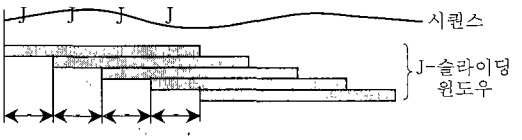


그림 1 J-슬라이딩 윈도우를 구성하는 방법

보조정리 5 시퀀스 S를 J-슬라이딩 윈도우로 나누었을 때, S의 서브시퀀스 S[i:j]에 포함된 첫번째 J-슬라이딩 윈도우는 S의 $\lfloor (i-1)/J \rfloor + 1$번째 J-슬라이딩 윈도우이다.

증명: 부록 A 참조

GeneralMatch에서는 J-슬라이딩 윈도우의 J를 ω의 약수로 한정한다. 이는 보조정리 1을 사용하기 위한 것으로, 보조정리 1을 사용하면 두 시퀀스의 디스조인트 윈도우 쌍을 비교하여 후보집합을 구할 수 있다. 시퀀스 S를 J-슬라이딩 윈도우로 나누었을 때, S[i:j]에 포함된 첫번째 윈도우가 s_a^J라 하자. 이때, J가 ω의 약수라면, s_a^J(시작 위치 = $(a-1)*J+1$)를 ω만큼 이동한 윈도우(시작 위치 = $(a-1)*J+1+\omega$)는 S의 $a+\omega/J$번째 J-슬라이딩 윈도우인 $s_{a+\omega/J}^J$가 된다. 따라서, S[i:j]의 서브시퀀스 $S[(a-1)*J+1:j]$의 디스조인트 윈도우들은 S의 J-슬라이딩 윈도우인 s_a^J, $s_{a+\omega/J}^J$, ...와 동일하고, 이들 J-슬라이딩 윈도우를 보조정리 1의 디스조인트 윈도우들로 이용할 수 있다. J가 ω의 약수라는 조건은 정리 1에서 사용되며, 본 논문에서는 설명의 편의를 위하여 ω/J를 k로 표기한다.

J-디스조인트 윈도우

GeneralMatch는 질의 시퀀스를 다음과 같이 정의되는 J-디스조인트 윈도우로 나눈다.

정의 2 시퀀스 Q의 크기 ω인 J-디스조인트 윈도우($1 \le J \le \omega$)는 시작 엔트리가 $Q[j+(j-1)*\omega]$($1 \le i \le J$, $1 \le j \le \frac{Len(Q)-i+1}{\omega}$)인 길이 ω의 서브시퀀스로 정의한다. 그리고, S를 J-디스조인트 윈도우로 나눈다함은 S의 모든 가능한 위치에 대해 J-디스조인트 윈도우를 구성함을 의미한다.

직관적으로는, 그림 2와 같이 Q[1:Len(Q)]를 디스조인트 윈도우로 나누어 $Q[1:\omega]$,$Q[1+\omega:2\omega]$,...의 윈도우를 구성하고, Q[2:Len(Q)]를 디스조인트 윈도우로 나누어 $Q[2:1+\omega]$,$Q[2+\omega:1+2\omega]$,...의 윈도우를 구성하며, 이를 반복하여 Q[J:Len(Q)]에 대해서는 $Q[J:J-1+\omega]$, $Q[J+\omega:J-1+2\omega]$,...의 윈도우를 구성하는 방법이다. 본 논문에서는 Q[i:Len(Q)]를 디스조인트 윈도우

로 나누었을 때 j 번째 윈도우인 Q[i+(j−1)*ω:i−1+j*ω]를 $q^J_{(i,j)}$로 나타낸다. 예를 들어, 그림 2에서 $q^J_{(1,1)}$과 $q^J_{(1,2)}$는 Q[1:Len(Q)]의 첫번째 및 두번째 디스조인트 윈도우이고, $q^J_{(2,1)}$과 $q^J_{(2,2)}$는 Q[2:Len(Q)]의 첫 번째 및 두 번째 디스조인트 윈도우를 나타낸다.

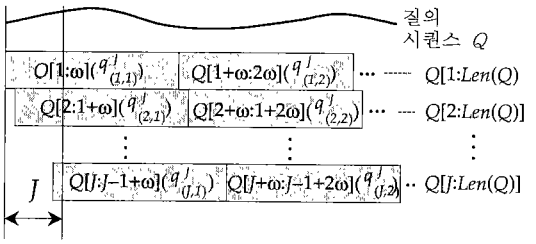


그림 2 질의 시퀀스를 J-디스조인트 윈도우로 나누는 방법

FRM과 DualMatch는 GeneralMatch의 특수한 예에 해당한다. 즉, 데이타 시퀀스를 슬라이딩 윈도우로 나누고 질의 시퀀스를 디스조인트 윈도우로 나누는 FRM은 GeneralMatch에서 J 값으로 1을 사용한 예에 해당한다. 그리고, 데이타 시퀀스를 디스조인트 윈도우로 나누고 질의 시퀀스를 슬라이딩 윈도우로 나누는 DualMatch는 GeneralMatch에서 J 값으로 ω를 사용한 예에 해당한다. 결과적으로, FRM은 데이타 시퀀스를 1-슬라이딩 윈도우로 나누고 질의 시퀀스를 1-디스조인트 윈도우로 나누는 방법을 사용하며, DualMatch는 데이타 시퀀스를 ω-슬라이딩 윈도우로 나누고 질의 시퀀스를 ω-디스조인트 윈도우로 나누는 방법을 사용한다. 따라서, 본 논문에서 제안하는 GeneralMatch는 윈도우 구성 방법에 있어서 기존의 FRM과 DualMatch를 아우르는 일반화된 서브시퀀스 매칭 방법이다.

GeneralMatch의 정확성

GeneralMatch는 다음 정리 1에 기반하여 착오기각 없이 서브시퀀스 매칭을 수행한다.

정리 1 데이타 시퀀스 S를 크기 ω인 J-슬라이딩 윈도우로 나누고 질의 시퀀스 Q를 같은 크기의 J-디스조인트 윈도우로 나누었을 때, 길이 Len(Q)인 S의 서브시퀀스 S[i:j]와 Q가 ε-매치하면, S[i:j]에 포함된 적어도 하나 이상의 J-슬라이딩 윈도우 $s^J_{a+n*k}(0 \le n \le \rho-1, k=\omega/J)$와 Q에 포함된 J-디스조인트 윈도우 $q^J_{(b+1,n+1)}$이 $\varepsilon/\sqrt{\rho}$-매치한다. 즉, 다음 조건식 (3)이 성립한다.

$$D(S[i:j],Q) \le \varepsilon \Rightarrow \bigvee_{n=0}^{\rho-1} D(s^J_{a+n*k}, q^J_{(b+1,n+1)}) \le \varepsilon/\sqrt{\rho} \qquad (3)$$

여기에서, $a = \lfloor (i-1)/J \rfloor + 1$이고, $b = (a-1)*J-i+1$이며, $\rho = \lfloor (Len(Q)-b)/\omega \rfloor$이다.

증명: 부록 B 참조

정리 1에 의해서, 질의 시퀀스 Q를 나눈 J-디스조인트 윈도우 $q^J_{(x,y)}$와 데이타 시퀀스 S를 나눈 J-슬라이딩 윈도우 s^J_z가 $\varepsilon/\sqrt{\rho}$-매치(정리 1에서 b+1이 x이므로, $\rho = \lfloor (Len(Q)-x+1)/\omega \rfloor$)할 때, 즉, 식 (3)의 필요조건을 만족할 때, S[(z−1)*J − (y−1)*ω−x+2][2]를 시작 엔트리로 하는 길이 Len(Q)인 서브시퀀스로 후보집합을 구성하면 착오기각 없이 모든 유사 서브시퀀스를 찾을 수 있다.

3.2 색인 구성 알고리즘

GeneralMatch의 색인 구성 알고리즘은 그림 3과 같다. 알고리즘 Build_Index에서는 데이타 시퀀스로 구성된 데이타베이스를 입력으로 받아, 서브시퀀스 매칭에 사용할 f-차원 색인을 구성한다. 알고리즘의 스텝 2.1에서는 식별자가 *S-id*인 각 데이타 시퀀스 S를 J-슬라이딩 윈도우로 나눈다. 스텝 2.2.1에서는 각 J-슬라이딩 윈도우 s^J_z를 f-차원 점인 *f-point*로 변환한다. 스텝 2.2.2에서는 변환된 점인 *f-point*, 데이타 시퀀스 식별자인 *S-id*, 그리고 S의 몇 번째 J-슬라이딩 윈도우인지를 나타내는 z로 레코드를 구성한다. 여기에서, 식별자 *S-id*는 색인을 검색한 후에 후보 서브시퀀스가 속한 데이타 시퀀스를 액세스하기 위해 사용하며, z는 해당 서브시퀀스의 위치를 알아내는데 사용한다.

알고리즘 **Build_Index**
입력: 데이타 시퀀스로 구성된 데이타베이스 *db*
출력: 서브시퀀스 매칭에 사용할 *f*-차원 색인 *index*
알고리즘:
1. *f*-차원 색인 *index*를 초기화 한다.
2. 데이타베이스 *db*에 포함된 각 데이타 시퀀스 S(식별자 *S-id*)에 대해서 다음을 수행한다.
   2.1 데이타 시퀀스 *S*를 *J*-슬라이딩 윈도우로 나눈다.
   2.2 각 J-슬라이딩 윈도우 s^J_z에 대해서 다음을 수행한다.
      2.2.1 특성 추출 함수를 사용하여 윈도우 s^J_z를 *f*-차원의 점인 *f-point*로 변환한다.
      2.2.2 변환한 점과 윈도우 s^J_z의 정보를 사용하여 <*f-point*, *S-id*, *z*>의 레코드를 구성한다.
      2.2.3 구성한 레코드를 *f-point*를 키로 하여 *index*에 삽입한다.

그림 3 색인 구성 알고리즘 Build_Index

2) 시작 엔트리의 위치(=i)는 정리 1에 따라, b=(a−1)*J−i+1에 b+1=x, n+1=y, 그리고 a+n*k=z을 적용하여 구할 수 있다.

알고리즘 Build_Index에서는 점 여과 효과를 얻기 위하여 J-슬라이딩 윈도우를 변환한 개별 점을 직접 색인에 저장하는 방법을 사용한다. 그러나, FRM과 같이 J 값이 작은 경우에는 색인에 저장하는 점의 개수가 너무 많아지므로[2], 이 방법을 그대로 적용하기는 어렵다. 반면에, DualMatch와 같이 J 값이 큰 경우에는 색인에 저장하는 점의 개수가 적으므로[4], 개별 점을 직접 색인에 저장할 수 있다. 이와 관련하여, GeneralMatch에서 색인에 저장하는 점의 개수에 대해서는 다음 제3.4절에서 자세히 설명한다. 본 논문에서는 FRM에서 개별 점을 직접 저장하는 경우를 *FRM-POINT*라 부르고, MBR을 구성하여 저장하는 기존 방법은 이와 구분하여 *FRM-MBR*이라 부르기로 한다.

3.3 서브시퀀스 매칭 알고리즘

GeneralMatch의 서브시퀀스 매칭 알고리즘은 그림 4와 같다. 알고리즘 General_Match에서는 시계열 데이타베이스, f-차원 색인, 질의 시퀀스 Q, 그리고 허용치 ε을 입력으로 받아, Q와 ε-매치하는 서브시퀀스를 포함하는 데이타 시퀀스와 해당 서브시퀀스의 시작 위치를 출력한다. General_Match는 크게 초기화, 색인 검색, 그리고 후처리의 세 단계로 구성된다.

첫째, 초기화 단계에서는 질의에 사용할 MBR을 구성한다. 우선, 질의 시퀀스를 J-디스조인트 윈도우로 나누어 f-차원 점으로 변환한다. 그리고, 범위 질의의 횟수를 줄이기 위하여 여러 개의 점들을 묶어서 MBR을 구성한다. MBR을 구성하는 방법은 1) FRM에서 사용한 휴리스틱 방법, 2) 고정 개수의 점으로 MBR을 구성하는 방법, 그리고 3) 모든 점을 하나의 MBR에 포함시키는 방법 등을 생각할 수 있다. 그러나, 실제 실험 결과 MBR 개수가 1~16인 경우, 이의 변화에 따른 성능 차이는 매우 작게 나타났다(11% 이하). 따라서, 본 논문에서는 문제를 단순화하기 위하여 세번째 방법인 하나의 MBR을 구성하는 방법을 사용한다.

```
알고리즘 General_Match
입력: (1) 데이타 시퀀스로 구성된 데이타베이스 db
      (2) 알고리즘 Build_Index로 구성한 색인 index
      (3) 질의 시퀀스 Q와 허용치 ε
출력: Q와 ε-매치하는 서브시퀀스를 포함하는 데이타 시퀀스의 식별자와 해당 서브시퀀스의 시작 위치
알고리즘:
1. 초기화 단계
   1.1 Q를 J-디스조인트 윈도우로 나누고, 각 윈도우를 f-차원 점으로 변환한다.
   1.2 변환한 점들로 MBR을 구성한다.
2. 색인 검색 단계: 각 MBR에 대해서 다음을 수행한다.
   2.1 MBR과 ε/√ρ으로 범위 질의를 구성하여 index를 검색한다.
   2.2 색인 수준 여과를 수행하여 ε/√ρ-매치하는 점의 레코드로 후보집합을 구성한다.
3. 후처리 단계: 후보집합에 포함된 각 레코드에 대해서 다음을 수행한다.
   3.1 각 레코드에 해당하는 후보 서브시퀀스를 db로부터 읽어 온다.
   3.2 읽어온 후보 서브시퀀스와 Q와의 Len(Q)-차원 거리 계산을 수행하여, 착오해답을 제거하고 유사(ε-매치하는) 서브시퀀스만을 찾는다.
```

그림 4 서브시퀀스 매칭 알고리즘 General_Match

둘째, 색인 검색 단계에서는 색인을 검색하여 후보집합을 구성한다. 우선, 스텝 2.1에서 MBR과 허용치 $\varepsilon/\sqrt{\rho}$으로 범위 질의를 구성하여 f-차원 색인을 검색한다. 그리고, 스텝 2.2에서 색인 수준 여과를 통하여 후보집합을 구성한다. 즉, 색인 검색 결과 얻은 각 점과 MBR에 포함된 각 점간의 거리를 계산하여 $\varepsilon/\sqrt{\rho}$-매치하는 점의 레코드를 해당 J-디스조인트 윈도우의 번호 <x, y>와 함께 후보집합에 포함시킴에 포함시킨다. 이와 같이 색인 수준 여과를 사용하는 이유는 개별 점 대신 MBR로 질의함으로써 증가할 수 있는 착오해답을 제거하고 개별 점을 사용한 경우와 같이 점 여과 효과를 발휘하기 위해서다[4].

셋째, 후처리 단계에서는 후보집합에 포함된 착오해답을 제거하고 유사 서브시퀀스만을 찾는다. 우선, 스텝 3.1에서 후보집합의 각 레코드 <*f-point, S-id, z*>에 대한 후보 서브시퀀스 *sub-S*를 데이타베이스로부터 읽어온다. 이는 *S-id*와 *sub-S*의 시작 위치(=(z−1)*J−(y−1)*ω−x+2)를 사용하여 수행된다. 여기에서, *S-id*와 *z*는 레코드에 포함되어 있으며, *x*와 *y*는 스텝 2.2에서 저장한 윈도우 번호이다. 스텝 3.2에서는 *sub-S*와 질의 시퀀스 Q와의 Len(Q)-차원 거리 계산을 수행하여, 착오해답을 제거하고 ε-매치하는 서브시퀀스를 찾는다.

3.4 최대 윈도우 크기와 색인에 저장하는 점의 개수

다음 보조정리 6은 GeneralMatch의 최소 질의 시퀀스 길이와 최대 윈도우 크기의 관계를 나타낸다.

보조정리 6 주어진 최소 질의 시퀀스 길이를 Min(Q)라 하면, 착오기각 없이 모든 유사 서브시퀀스를 찾아내기 위한 GeneralMatch의 최대 윈도우 크기는 ⌊(Min (Q)−J+1)/J⌋*J이다.

증명: 부록 C 참조

GeneralMatch에서 색인에 저장되는 점의 개수는 다음 보조정리 7을 사용하여 구할 수 있다.

보조정리 6 GeneralMatch에서 데이타 시퀀스 S를 크기 ω인 J-슬라이딩 윈도우로 나누는 경우, 색인에 저장되는 점의 개수는 ⌊(Len(S)−ω)/J⌋+1이다.

증명: 부록 D참조

표 2는 최소 질의 시퀀스 길이 Min(Q)가 512이고 Len(S)가 1000000인 경우에 J 값의 변화에 따른 최대 윈도우 크기와 색인에 저장하는 점의 개수를 나타낸다. 표에서 알 수 있듯이, J값이 클수록 최대 윈도우 크기는 작아지고 색인에 저장해야 하는 점의 개수는 줄어든다. 결국, J 값이 클수록 윈도우 크기 효과에 의한 착오해답이 증가하여 데이타베이스에 대한 디스크 I/O가 늘어나는 반면에, 색인에 저장해야 하는 점의 개수가 줄어들어 색인을 위한 저장 공간과 색인에 대한 디스크 I/O가 줄어든다. 데이타베이스 응용은 일반적으로 디스크 I/O에 의해 그 성능이 결정되므로, 다음 제3.5절에서는 디스크 I/O, 즉 페이지 액세스 횟수를 최소화하도록 J 값을 결정하는 방법을 제안한다.

표 2 Min(Q)=512, Len(S)=1000000인 경우의 최대 윈도우 크기와 색인에 저장하는 점의 개수

J	k(=ω/J)	ω	# of points	비 고
1	512	512	999489	FRM-POINT (J=1)
2	255	510	499746	–
3	170	510	333164	–
⋮	⋮	⋮	⋮	–
128	3	384	7810	–
171	2	342	5375	–
256	1	256	3906	DualMatch (J=ω)

3.5 최적 J 값의 결정 방법

본 논문에서는 서브시퀀스 매칭 과정에서 발생하는 페이지 액세스 횟수(= 색인 페이지 액세스 횟수 + 데이타 페이지 액세스 횟수)를 추정하여, 이를 최소화하는 방향으로 J 값을 결정한다. 그런데, 페이지 액세스 횟수는 질의 시퀀스와 허용치의 쌍 (Q, ε)에 따라 달라지므로, 본 논문에서는 여러 개의 샘플 (Q, ε) 쌍이 주어진다고 가정하고, 이들에 대한 평균 페이지 액세스 횟수가 최소화되도록 J 값을 결정한다. 각 J에 대한 평균 페이지 액세스 횟수를 계산하기 위해서는 표 3의 표기법을 사용한다. 제안하는 방법은 각 J에 대해서 $n_{page}(\mathbf{Q})$를 추정하고, 이 값이 가장 작은 J를 GeneralMatch에 사용한다. 이를 위하여, 우선 색인 페이지 액세스 횟수와 데이타 페이지 액세스 횟수의 추정하고, 이를 사용하여 평균 페이지 액세스 횟수를 구한다.

$n_{ipage}(Q, \varepsilon)$: 색인 페이지 액세스 횟수

표 3 평균 페이지 액세스 횟수 계산을 위한 표기법

기호	정의/의미
Q	샘플로 주어지는 질의 시퀀스와 허용치의 쌍 (Q, ε)들의 집합
$n_{page}(\mathbf{Q})$	**Q**의 모든 (Q, ε)에 대한 평균 페이지 액세스 횟수
$n_{ipage}(Q, \varepsilon)$	(Q, ε)에 대한 색인 페이지 액세스 횟수
$n_{dpage}(Q, \varepsilon)$	(Q, ε)에 대한 데이타 페이지 액세스 횟수
$size_i$	구성될 색인의 크기 (단위: 페이지)
$size_d$	시계열 데이타베이스의 크기 (단위: 페이지)
f_{int}	색인의 내부(internal) 페이지의 평균 팬-아웃
f_{leaf}	색인의 단말(leaf) 페이지의 평균 팬-아웃
n_{ipts}	색인에 저장될 점의 개수
$n_{pts}(Q, \varepsilon)$	(Q, ε)에 대한 범위 질의로 색인을 검색할 때, 색인에서 검색되는 점의 개수
$n_{subs}(Len(Q))$	길이 Len(Q)인 모든 가능한 데이타 서브시퀀스 개수
$n_{cand}(Q, \varepsilon)$	(Q)에 대해 발생하는 후보 서브시퀀스 개수
h	구성될 색인의 높이

각 J에 대한 (Q, ε)의 색인 페이지 액세스 횟수는 다음과 같이 추정한다[3]. 본 논문의 GeneralMatch에서는 각 (Q, ε) 당 한 번의 범위 질의를 수행한다. 그리고, 주어진 (Q, ε)에 대한 색인 검색 결과로는 색인에 저장된 총 n_{ipts}개의 점 중에서 $n_{pts}(Q, \varepsilon)$의 점을 찾아낸다. 따라서, (Q, ε)에 대한 색인 검색에 있어서는 전체 색인 페이지 중에서 대략 $\frac{n_{pts}(Q, \varepsilon)}{n_{ipts}}$(저장된 점의 개수에 대한 검색 결과 얻은 점의 개수 비율)만큼을 액세스한다고 가정한다. 결국, 색인 페이지 액세스 횟수는 다음 식 (4)와 같이 추정할 수 있다.

$$n_{ipage}(Q, \varepsilon) \approx size_i \times \frac{n_{pts}(Q, \varepsilon)}{n_{ipts}} \quad (4)$$

식 (4)에서 색인 크기 $size_i$는 색인에 저장된 점의 개수와 색인을 구성하는 페이지들의 팬-아웃인 f_{int}과 f_{leaf}에 의해 다음 식 (5)와 같이 추정할 수 있다. 식 (5)에서 $\left\lceil \frac{n_{ipts}}{f_{leaf}} \right\rceil$는 단말 페이지의 개수를, $\left\lceil \frac{n_{ipts}}{f_{leaf} \times f_{int}^{k}} \right\rceil (1 \le k \le h-1)$는 내부 페이지의 개수를 추정한 것이다.

3) Faloutsos와 Kamel[18]은 프랙탈 차원(fractal dimension)의 개념을 사용하여 R^*-트리에서의 색인 페이지 액세스 횟수를 추정하였으나, 본 논문에서는 이 추정 방법을 사용치 않았다. 그 이유는 1) 서브시퀀스 매칭에서 윈도우들이 변환된 f-차원 점들의 집합이 프랙탈이라는 사실이 밝혀진 바 없으며, 2) 참고문헌 [18]에서는 각 단계(level)의 MBR들이 크기가 동일한 정사각형 모양(square-like MBRs of the same size)임을 가정하였으나 실제로 f-차원 점들을 저장한 R^*-트리를 분석한 결과 MBR은 다양한 모양의 직사각형이었기 때문이다.

$$size_i \approx \left\lceil \frac{n_{ipts}}{f_{leaf}} \right\rceil + \left\lceil \frac{n_{ipts}}{f_{leaf} \times f_{int}} \right\rceil + \cdots + \left\lceil \frac{n_{ipts}}{f_{leaf} \times f_{int}^{h-1}} \right\rceil = \sum_{k=0}^{h-1} \left\lceil \frac{n_{ipts}}{f_{leaf} \times f_{int}^{k}} \right\rceil \quad (5)$$

그러면, 식 (5)를 식 (4)에 대입하여 $n_{ipage}(Q, \varepsilon)$을 다음 식 (6)과 같이 구할 수 있다.

$$n_{ipage}(Q,\varepsilon) \approx \left(\sum_{k=0}^{h-1} \left\lceil \frac{n_{ipts}}{f_{leaf} \times f_{int}^{k}} \right\rceil \right) \times \frac{n_{pts(Q,\varepsilon)}}{n_{ipts}} \quad (6)$$

$n_{dpage}(Q, \varepsilon)$: 데이타 페이지 액세스 횟수

각 J에 대한 (Q, ε)의 데이타 페이지 액세스는 후처리 과정에서 후보 서브시퀀스를 액세스하면서 발생한다. 그런데, 서브시퀀스 매칭에서는 많은 후보 서브시퀀스들이 인접하여 위치하고, 또한 동일한 페이지에 저장되어 한꺼번에 액세스될 가능성이 크다[4]. 따라서, (Q, ε)에 대한 데이타 페이지 액세스 횟수 $n_{dpage}(Q, \varepsilon)$은 다음 식 (7)과 같이 추정할 수 있다. 즉, 전체 데이타 페이지 중에서 대략 $\frac{n_{cand}(Q, \varepsilon)}{n_{subs}(Len(Q))}$(모든 서브시퀀스 개수에 대한 후보 서브시퀀스 개수 비율) 만큼의 페이지를 액세스한다고 볼 수 있다.

$$n_{dpage}(Q,\varepsilon) \approx size_d \times \frac{n_{cand}(Q, \varepsilon)}{n_{subs}(Len(Q))} \quad (7)$$

$n_{page}(Q)$: 평균 페이지 액세스 횟수

각 J에 대한 $n_{page}(\mathbf{Q})$는 $\mathbf{Q}$에 포함된 (Q, ε)들에 대한 $n_{ipage}(Q, \varepsilon)$와 $n_{dpage}(Q, \varepsilon)$을 사용하여 식 (8)과 같이 표현된다. 그리고, 식 (6)과 (7)을 식 (8)에 대입하여 $n_{page}(\mathbf{Q})$을 식 (9)와 같이 구할 수 있다.

$$n_{page}(\mathbf{Q}) = average[(n_{ipage}(Q, \varepsilon) + n_{dpage}(Q, \varepsilon))], \quad for(Q, \varepsilon) \in \mathbf{Q} \quad (8)$$

$$n_{page}(\mathbf{Q}) \approx average \left[\left(\left(\sum_{k=0}^{h-1} \left\lceil \frac{n_{ipts}}{f_{leaf} \times f_{int}^{k}} \right\rceil \right) \times \frac{n_{pts}(Q,\varepsilon)}{n_{ipts}} + size_d \times \frac{n_{cand}(Q,\varepsilon)}{n_{subs}(Len(Q))} \right) \right], \quad for(Q,\varepsilon) \in \mathbf{Q} \quad (9)$$

페이지 액세스 횟수 계산을 위한 변수 값의 결정

평균 페이지 액세스 횟수를 추정한 식 (9)에는 데이타베이스 액세스 없이 그 값을 알 수 있는 변수와 그렇지 않은 변수가 있다. 우선, f_{int}과 f_{leaf}은 색인의 종류, 페이지 크기, 특성의 개수 등에 의해서 알 수 있으며, $size_d$는 데이타베이스 크기에 의해 알 수 있다. 그리고, h $\left[= 1 + \lceil \log_{f_{leaf}} \frac{n_{ipts}}{f_{int}} \rceil \right]$에 포함된 n_{ipts}는 보조정리 6에 의해 계산할 수 있으며, $n_{subs}(Len(Q))$는 데이타 시퀀스의 개수와 길이를 사용하여 구할 수 있다. 즉, 이들 다섯 가지 값은 데이타베이스를 직접 액세스하지 않고도 알 수 있다. 그러나, $n_{pts}(Q, \varepsilon)$과 $n_{cand}(Q, \varepsilon)$은 데이타 및 질의 시퀀스의 특성, ε에 따른 범위 질의 영역, 윈도우 크기 효과 등의 복합적 요소에 의해서 결정되는 사항이므로, 데이타베이스 액세스 없이 그 값을 예측하기가 어렵다. 따라서, 본 논문에서는 한 번의 데이타베이스 스캔을 통하여 모든 $(Q, \varepsilon)(\in \mathbf{Q})$의 모든 J에 대한 $n_{pts}(Q, \varepsilon)$과 $n_{cand}(Q, \varepsilon)$의 정확한 값을 알아내는 방법을 사용한다.

4. 성능 평가

본 절에서는 FRM-POINT, FRM-MBR, DualMatch, 그리고 GeneralMatch의 성능 평가 결과를 설명한다. 제4.1절에서는 성능 평가를 수행한 실험 데이타와 실험 환경을 소개하고, 제4.2절에서는 실험 결과를 설명한다.

4.1 실험 데이타 및 실험 환경

실험 데이타로는 실제 주식 데이타와 합성 데이타(synthetic data)를 사용하였다. 사용한 데이타는 하나의 긴 데이타 시퀀스로 구성된 것으로서, 이는 여러 개의 데이타 시퀀스로 구성된 경우와 동일한 효과를 가진다. 주식 데이타[4]는 329112개의 엔트리로 구성되어 있으며[2, 4], 이를 *STOCK-DATA*라 한다. 합성 데이타는 100만 개의 엔트리로 구성된 유사 주기를 가지는 시계열 데이타(pseudo periodic synthetic time-series data)[5]로서, 이를 *PERIODIC-1M*(M은 million을 의미함)이라 한다. 확장성(scalability) 실험을 위해서는, 시작 엔트리를 1.5로 하고, 각 엔트리에 (−0.001, 0.001) 사이의 임의의 값 하나를 더하여 다음 엔트리를 구하는 방식으로 생성된 100만개 엔트리의 랜덤 워크 데이타(random walk data)인 *WALK-1M*과 이를 10번 반복하여 생성한 1000만개 엔트리의 *WALK-10M*, 100번 반복하여 생성한 1억개 엔트리의 *WALK-100M*을 사용하였다.

실험은 512M 바이트 메모리를 가진 SUN Ultra 60 워크스테이션에서 수행하였다. UNIX 시스템 자체의 버퍼링 효과를 피하고 실제 디스크 I/O를 보장하기 위하

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

5) 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.

여 데이타 및 색인 파일에 대해 원시 디스크(raw disk)를 사용하였으며, 데이타 및 색인 페이지의 크기는 4096 바이트로 하였다. 다차원 색인으로는 네 가지 방법 모두 R^*-트리[12]를 사용하였으며, GeneralMatch에서 J 값을 결정할 때 필요한 R^*-트리의 페이지 사용율은 일반적인 값인 69%로 하였다. 특성 추출 함수로는 DFT 변환을 사용하였으며, 특성은 6개[6]를 사용하였다.

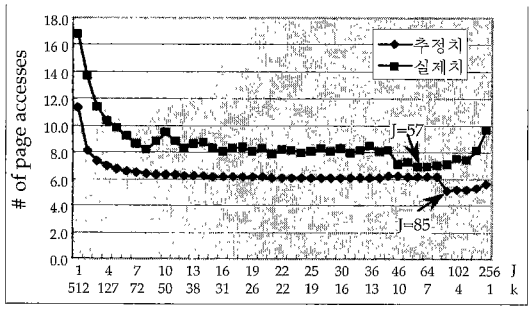
실험 결과로는 각 방법의 후보 개수, 페이지 액세스 횟수, 그리고 수행 시간을 측정하였다. 질의 시퀀스는 데이타 시퀀스의 임의 위치(random offset)를 시작 엔트리로 하는 길이 Len(Q)의 서브시퀀스로 하였으며, 질의 시퀀스의 길이는 512~1024로 하고, 각 길이당 10개씩의 질의 시퀀스를 생성하였다. 질의에 대한 선택률은 Low-Range(10^{-6}~10^{-4})와 High-Range(10^{-3}~10^{-1})의 두 가지 범위를 사용하였다. 그리고, 원하는 선택률은 각 질의에 대해서 허용치 ε을 조절하여 구하였으며, 각 범위 내에서는 선택률이 골고루 분포하도록 하였다.

4.2 실험 결과

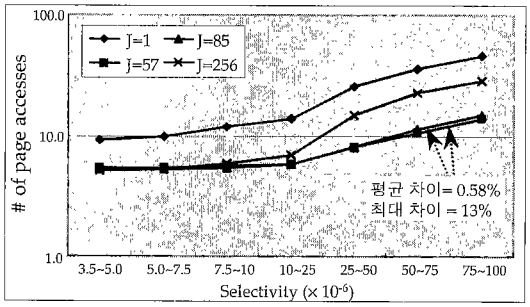
본 절에서는 STOCK-DATA에 대한 실험 결과를 먼저 설명한 후, PERIODIC-1M과 확장성 실험을 위한 WALK-1M/10M/100M에 대한 실험 결과를 소개한다.

STOCK-DATA의 실험 결과

우선, 실험에 사용할 질의 시퀀스의 3%를 임의로 선택하여, 최적 J 값 추정 방법의 정확성을 실험하였다. 각 J에 대한 평균 페이지 액세스 횟수의 추정치는 식 (9)를 사용하여 구하였고, 실제치는 각 J에 대해 실제 색인을 구성한 후 서브시퀀스 매칭을 수행하여 구하였다. 그림 5(a)는 Low-Range인 경우 각 J에 대한 평균 페이지 액세스 횟수의 추정치와 실제치를 보여준다. 실험 결과, 최적 J 값의 추정치는 85이고 실제치는 57로 나타났다. 그림 5(a)의 실제치를 보면, J 값이 57~85인 경우는 페이지 액세스가 매우 유사함을 알 수 있다. 그림 5(b)는 J 값으로 85(= 추정치), 57(= 실제치), 256 (DualMatch인 경우), 그리고 1(FRM-POINT인 경우)을 사용하여 Low-Range에 대한 페이지 액세스 횟수를 측정한 실험 결과이다. 그림을 보면, 추정치와 실제치를 사용한 경우가 다른 두 극단값인 256이나 1을 사용한 경우보다 페이지 액세스 횟수를 줄임을 알 수 있다. 반면에, 추정치와 실제치의 경우 페이지 액세스 횟수가 거의 유사함을 알 수 있다. 실제로 두 값을 사용한 페이지 액세스 횟수의 차이는 평균 0.53%, 최대 13%에 불과하였다. 이같은 결과로 볼 때, 본 논문에서 제안한 최적 J 값의 추정 방법은 비교적 정확한 추정을 한다고 말할 수 있다.



(a) J 값의 변화에 따른 추정치와 실제치 비교

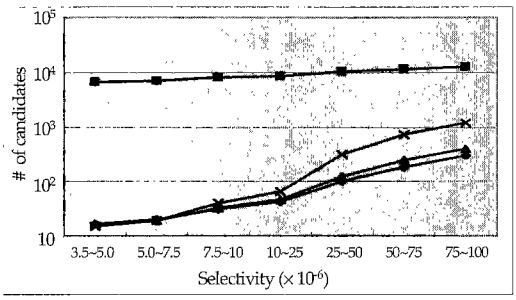


(b) 추정치와 실제치를 사용한 페이지 액세스 횟수 비교

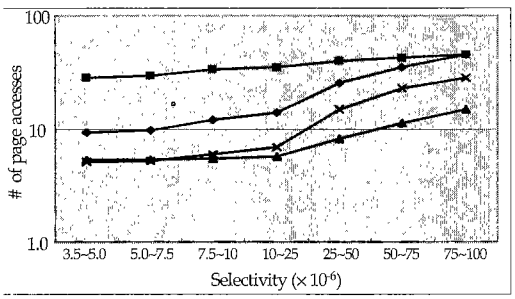
그림 5 STOCK-DATA에 Low-Range인 경우의 최적 J 값의 추정치와 실제치 비교

그림 6은 STOCK-DATA에 Low-Range인 경우의 실험 결과를 나타낸다. 이 실험에서 GeneralMatch의 J 값으로는 추정치인 85(ω는 425)를 사용하였다. 그림을 보면, FRM-MBR은 DualMatch와 GeneralMatch는 물론 FRM-POINT보다도 세 가지 실험 결과 모두 좋지 않게 나타났는데, 이는 FRM-MBR에서 점 여과 효과가 결여되었기 때문이다. 그리고, 윈도우 크기 효과가 가장 크며 점 여과 효과를 발휘하는 FRM-POINT는 후보 개수는 가장 적으나, 색인 검색의 오버헤드로 인하여 페이지 액세스 횟수와 수행 시간은 GeneralMatch에 뒤지는 것으로 나타났다. 또한, GeneralMatch는 Dual

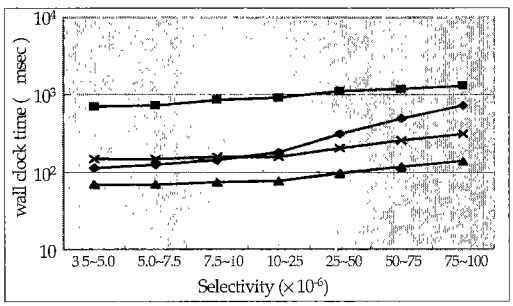
6) DFT 변환에서는 첫 번째 복소수의 허수부 0대신, 네 번째 복소수의 실수부를 사용하였다. 그리고, 실수에 대한 DFT 변환의 경우, 대칭 성질에 의하여 뒤쪽에 있는 계수들은 처음에 있는 계수들의 켤레복소수이므로, 마지막 몇 개의 특성 또한 처음 몇 개의 특성 만큼 에너지를 발휘한다[19]. 좀 더 정확히 이야기하면, i($2 \le i \le \omega$) 번째 복소수는 'ω-i+2'번째 복소수의 켤레복소수이다. 따라서, 두 번째~네 번째 복소수에 $\sqrt{2}$를 곱하여 특성을 추출하였는데, 이는 Rafiei와 Mendelzon [19]의 접근법인 질의 영역에 2대신 $\sqrt{2}$를 곱하는 것과 동일한 효과를 갖는다.



(a) 후보개수



(b) 페이지 엑세스 횟수



(c) 수행 시간

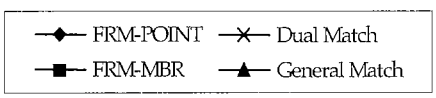


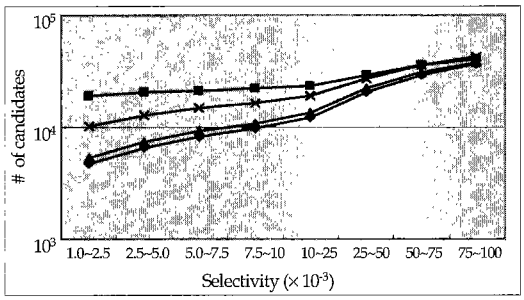
그림 6 STOCK-DATA에 Low-Range인 경우의 선택률에 따른 실험 결과

Match보다 약간 우수한 결과를 보이는데, 이는 General Match가 DualMatch보다 윈도우 크기 효과를 크게 발휘하기 때문이다. Low-Range의 실험 결과를 정리하면, GeneralMatch는 성능(수행시간)에 있어서 DualMatch와 FRM-MBR에 비해 각각 114%와 998%를 평균적으로 향상시킨 것으로 나타났다. 그리고, 후보 개수는 두 가지 방법에 비해 각각 86%와 20200%를 평균적으로 줄였으며, 페이지 액세스 횟수는 두 가지 방법에 비해 각각 44%와 408%를 평균적으로 줄인 것으로 나타났다.

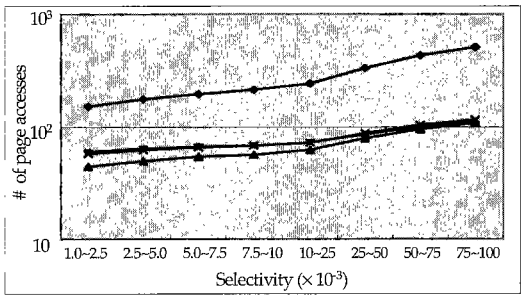
그림 7은 STOCK-DATA에 High-Range인 경우의 실험 결과를 나타낸다. 이 실험에서는 최적 J 값으로 57을(ω는 456) 사용하였다. 그림을 보면, FRM-POINT는 후보 개수는 가장 적으나, 색인 검색의 오버헤드로 인하여 페이지 액세스 횟수와 수행 시간에 있어서는 다른 세 가지 방법에 크게 뒤지는 것으로 나타났다. 그리고, 선택률이 높은 일부 구간에서는 DualMatch가 FRM-MBR보다 성능이 약간 저하되는데, 이는 선택률이 높아질수록 윈도우 크기 효과가 점 여과 효과보다 크게 나타나기 때문이다[4]. 반면에, GeneralMatch는 선택률이 높은 구간에서도 FRM-MBR보다 우수한 결과를 보이고 있다. 이는 GeneralMatch가 비교적 큰 윈도우(FRM의 89% (= 456/512))를 사용하면서 점 여과 효과를 발휘하기 때문이다. High-Range의 실험 결과를 정리하면, General Match는 성능에 있어서 DualMatch와 FRM- MBR에 비해 각각 46%와 65%를 평균적으로 향상시킨 것으로 나타났다. 그리고, 후보 개수는 두 가지 방법에 비해 각각 49%와 102%를 평균적으로 줄였으며, 페이지 액세스 횟수는 두 가지 방법에 비해 각각 18%와 18%를 평균적으로 줄인 것으로 나타났다.

PERIODIC-1M의 실험 결과

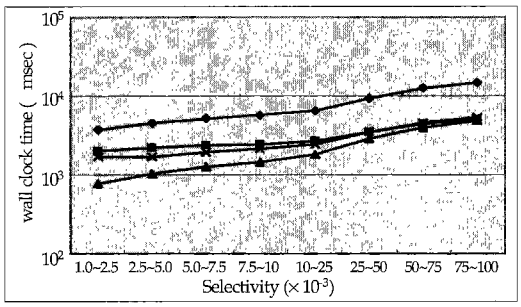
그림 8은 PERIODIC-1M에 Low-Range인 경우의 실험 결과를 나타낸다. 이 실험에서는 최적 J 값으로 57을 사용하였다. 그림을 보면, 세 가지 실험 결과 모두 그림 6의 실험 결과와 경향이 유사하다. 그러나, General Match와 FRM-MBR(혹은 DualMatch와 FRM-MBR)의 성능 차이는 더욱 크게 나타남을 알 수 있다. PERIODIC-1M은 비슷한 서브시퀀스가 큰 주기를 가지고 반복하여 나타나지만, STOCK-DATA에 비하여 인접한 엔트리들의 변화가 큰 특징을 가지고 있다. 따라서, 인접한 슬라이딩 윈도우들이라 할지라도 STOCK- DATA에 비하여 거리가 먼 경우가 많이 발생한다. 그러므로, MBR을 구성하여 저장하는 FRM-MBR에서는 동일한 MBR에 포함된 윈도우들이라 할지라도 윈도우들 사이의 거리가 먼 경우가 많아지고, 이들이 함께 후보집합에 포함됨으로써 많은 착오해답이 발생하게 된다. 반면에, GeneralMatch와 Dual Match에서는 MBR대신 개별 점을 저장 및 검색에 사용함으로써 이러한 문제가 발생하지 않는다. 이와 같은 이유에 의하여 PERIODIC-1M은 STOCK -DATA에 비하여 실험 결과가 큰 차이를 나타내게 되었다[4]. 그리고, 그림 8(b)를 보면 선택률이 7.5×10^{-6} 이하인 구간에서



(a) 후보개수



(b) 페이지 액세스 횟수



(c) 수행 시간

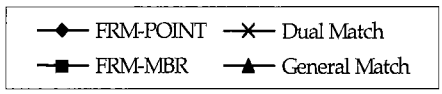
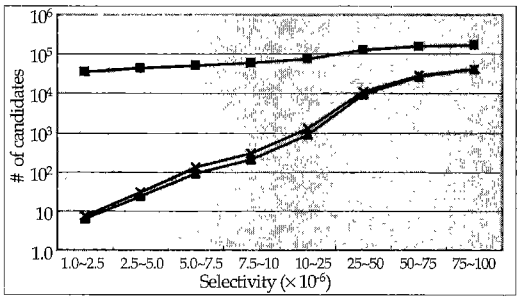
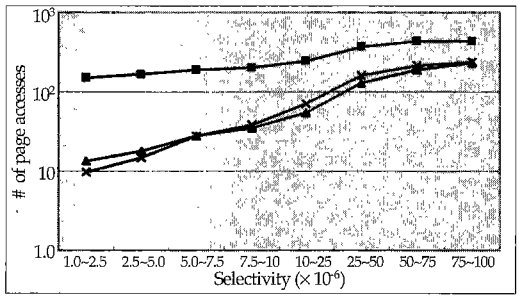


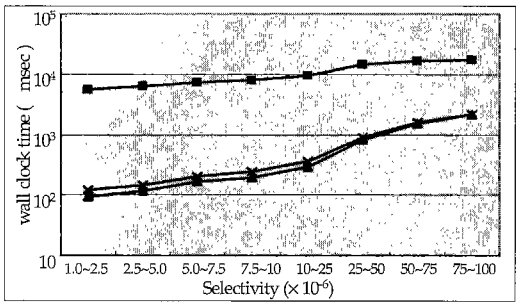
그림 7 STOCK-DATA에 High-Range인 경우의 선택률에 따른 실험 결과



(a) 후보개수



(b) 페이지 엑세스 횟수



(c) 수행 시간

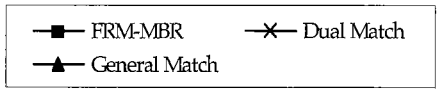
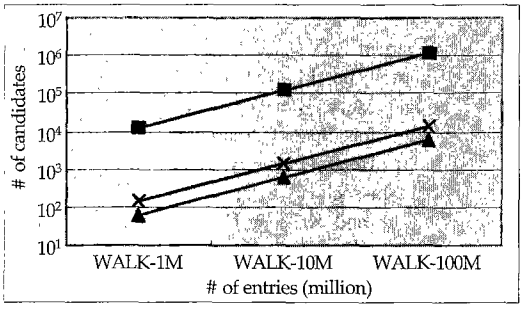


그림 8 PERIODIC-1M에 Low-Range인 경우의 선택률에 따른 실험 결과

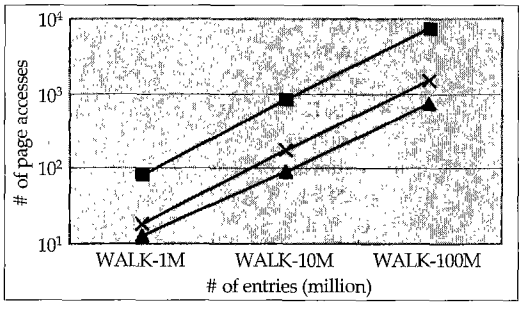
GeneralMatch가 Dual Match에 비해 페이지 액세스 횟수가 많은 것으로 나타났는데, 이는 J 값 결정시 1.0×10^{-6}~7.5×10^{-6}이 아닌 1.0×10^{-6}~1.0×10^{-4}를 선택률 범위로 하여 평균 페이지 액세스 횟수를 최소로 하는 J 값을 선택하였기 때문이다.

WALK-1M/10M/100M의 실험 결과

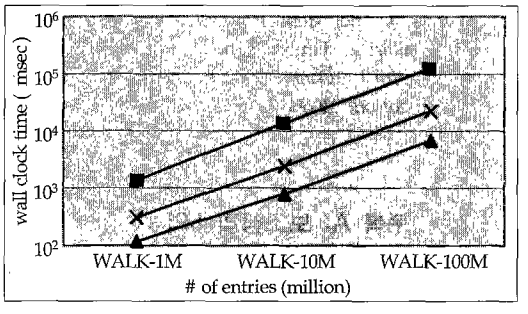
그림 9는 WALK-1M/WALK-10M/WALK-100M에 대해서, 선택률이 10^{-6}~10^{-5}인 경우의 실험 결과를 나타낸다. 이 실험에서는 최적 J 값으로 57을 사용하였다. 그림을 보면, GeneralMatch가 FRM-MBR은 물론 DualMatch에 비해 항시 우수하며, 데이타베이스의 크기가 증가하더라도 세 가지 방법의 후보 개수, 페이지



(a) 후보 개수



(b) 페이지 엑세스 횟수



(c) 수행 시간

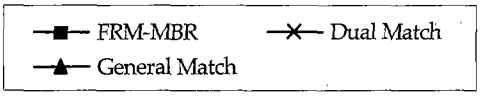


그림 9 선택률이 10^{-6}~10^{-5}인 경우 WALK-1M/10M /100M에 대한 실험 결과

액세스 횟수, 수행 시간의 차이가 일정 비율로 유지되는 것으로 나타났다. 결국, GeneralMatch는 대용량 환경하에서도 기존 방법들에 비해서 우수한 성능을 나타냄을 알 수 있다.

5. 결 론

본 논문에서는 윈도우 구성의 일반화 개념을 제안하고, 이에 기반하여 DualMatch를 더욱 개선한 서브시퀀스 매칭 방법인 GeneralMatch를 제안하였다. General Match는 슬라이딩 윈도우와 디스조인트 윈도우를 일반화한 J-슬라이딩 윈도우와 J-디스조인트 윈도우를 사용하여, 데이타 시퀀스를 J-슬라이딩 윈도우로 나누고 질의 시퀀스를 J-디스조인트 윈도우로 나누는 방법을 사용한다. 본 논문에서는 GeneralMatch의 정확성을 정리 1에서 증명하였다. 그리고, 주어진 최소 질의 시퀀스 길이에 대해 GeneralMatch가 바르게 동작하기 위한 최대 윈도우 크기를 보조정리 6에서 구하였다. 다음으로, 보조정리 7에서 GeneralMatch에서 색인에 저장하는 점의 개수를 구하였다. 마지막으로, 페이지 액세스 횟수를 최소로 하는 J 값의 결정 방법을 제안하였다.

본 논문에서는 데이타의 종류와 선택률 범위를 달리하면서 많은 실험을 수행하였다. 실험 결과, General Match는 데이타의 종류와 크기, 선택률 범위, 질의 시퀀스의 길이에 관계없이 FRM-MBR은 물론 Dual Match보다 후보집합 크기와 페이지 액세스 횟수를 줄이고 성능을 향상시켰다. 실제 주식 데이타에 대한 실험 결과, GeneralMatch는 낮은 선택률 범위(10^{-6}~10^{-4})에서 Dual Match에 비해 평균 114%, FRM에 비해 평균 998% 성능을 향상시켰으며, 높은 선택률 범위(10^{-3}~10^{-1})에서도 DualMatch에 비해 평균 46%, FRM에 비해 평균 65% 성능을 향상시켰다.

GeneralMatch는 데이타의 종류와 크기, 빈번하게 사용되는 질의의 종류, 선택률의 범위 등에 따라 적합한 J 값을 사용할 수 있다. 또한, FRM 및 DualMatch에 비해 항상 우수한 성능을 나타낸다. 이 같은 결과로 볼 때, GeneralMatch는 FRM 및 DualMatch에 비해 보다 다양한 응용에서 적합하게 사용될 수 있을 것으로 사료된다.

참 고 문 헌

[1] 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.

[2] 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.

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

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

[5] 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.

[6] 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(ICDE)*, IEEE, Orlando, Florida, pp. 201-208, Feb. 1998.

[7] 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.

[8] 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.

[9] 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 (ICDE)*, IEEE, San Diego, California, pp. 23-32, March 2000.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Faloutsos, C. and Kamel, I., "Beyond Uniformity and Independence: Analysis of R-trees Using the Concept of Fractal Dimension," In *Proc. the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Minneapolis, Minnesota, pp. 4-13, May 1994.

[19] Rafiei, D. and Mendelzon, A., "Efficient Retrieval of Similarity Time Sequences Using DFT," In *Proc. Int'l Conf. on Foundations of Data Organization*, Kobe, Japan, pp. 249-257, Nov. 1998.

부록 A: 보조정리 5의 증명

서브시퀀스 S[i:j]에 포함된 첫번째 J-슬라이딩 윈도우가 S의 a번째 J-슬라이딩 윈도우인 s_a^J라 하자. 그러면, s_a^J는 S[(a-1)*J+1:(a-1)*J+ω]이므로 다음 식 (10)이 성립한다.

$$(a-1)*J+1 \geq i \iff a \geq \frac{i-1}{J}+1 \tag{10}$$

그런데, a는 정수이므로 식 (10)에 의해 다음 식 (11)이 성립한다.

$$a \geq \left\lfloor \frac{i-1}{J} \right\rfloor + 1 \tag{11}$$

그리고, s_a^J가 S[i:j]에 포함된 첫번째 J-슬라이딩 윈도우이므로 $a = \lfloor (i-1)/J \rfloor + 1$이 된다. 그 이유는 만일

$a > \lfloor (i-1)/J \rfloor + 1$이라면, a와 $\lfloor (i-1)/J \rfloor + 1$이 정수이므로 $(a-1) \geq \lfloor (i-1)/J \rfloor + 1$이 성립하고, 결국, s_a^J의 바로 이전 J-슬라이딩 윈도우인 $s_{(a-1)}^J$이 S[i:j]에 포함되기 때문이다. 따라서, S[i:j]에 포함된 첫번째 J-슬라이딩 윈도우는 $\lfloor (i-1)/J \rfloor + 1$번째 J-슬라이딩 윈도우이다.

부록 B: 정리 1의 증명

증명을 위해 그림 10을 사용한다. 우선, 서브시퀀스 S[i:j]에 포함된 첫번째 J-슬라이딩 윈도우를 S의 a 번째 J-슬라이딩 윈도우인 $s_a^J (= S[(a-1)*J+1:(a-1)*J+\omega])$라 하자. 그러면, S[i:j]는 $S[i:(a-1)*J]S[(a-1)*J+1:j]$로 나타낼 수 있고, 다시 그림 10과 같이 J-슬라이딩 윈도우들과 앞뒤의 서브시퀀스 s_h(h는 *head*를 의미함)와 s_t(t는 *tail*을 의미함)를 사용하여 $s_h\, s_a^J\, s_{a+k}^J \cdots s_{a+(\rho-1)*k}^J\, s_t$로 나타낼 수 있다. 이와 같이 나타낼 수 있는 이유는 J를 ω의 약수로 한정했기 때문이다. 다음으로, $Len(s_h)$를 b로 하여 Q를 Q[1:b]Q[b+1:Len(Q)]로 나타낼 수 있다. 그런데, s_a^J가 S[i:j]에 포함된 '첫번째' J-슬라이딩 윈도우이므로 $0 \leq Len(s_h) = b \leq J-1$이 성립하고, 이에 따라 $1 \leq b+1 \leq J$이 성립한다. 따라서, Q는 그림 10과 같이 Q[b+1]에서 시작하는 J-디스조인트 윈도우들인 $q_{(b+1,1)}^J$, $q_{(b+1,2)}^J$, …을 사용하여 $q_h\, q_{(b+1,1)}^J$, $q_{(b+1,2)}^J$, $\cdots q_{(b+1,\rho)}\; q_t (Len(q_h)=Len(s_h),\ Len(q_t)=Len(s_t))$로 나타낼 수 있다. 이와 같이 나타내면, 보조정리 1과 2에 의하여 다음과 같은 과정으로 식 (12)가 성립한다.

$$
\begin{aligned}
& D(s[i:j],Q) \leq \varepsilon \\
\Rightarrow\ & D(s_a^J \cdots s_{a+(\rho-1)*k}^J, q_{(b+1,1)}^J \cdots q_{(b+1,\rho)}^J) \leq \varepsilon \quad (\text{보조정리2}) \\
\Rightarrow\ & \bigvee_{n=0}^{\rho-1} D(s_{a+n*k}^J, q_{(b+1,n+1)}^J) \leq \varepsilon/\sqrt{\rho} \quad (\text{보조정리1}) \qquad (12) \\
& \cdots \qquad\qquad \cdots
\end{aligned}
$$

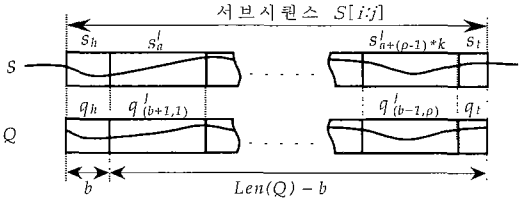


그림 10 J-슬라이딩 윈도우와 J-디스조인트 윈도우를 이용한 서브시퀀스와 질의 시퀀스의 표현

그리고, s_a^J가 S[i:j]의 첫번째 윈도우이므로 보조정리 3에 의하여 a는 $\lfloor (i-1)/J \rfloor + 1$이고, $Len(s_h)(=b)$는 Len(S[i:(a−1)*J])이므로 b는 $(a-1)*J-i-1$이며, Q[b+1:Len(Q)]는 $\lfloor (Len(Q)-b)/\omega \rfloor$개의 디스조인트 윈도우로 나누어지므로, ρ는 $\lfloor (Len(Q)-b)/\omega \rfloor$가 된다.

부록 C: 보조정리 6의 증명

정리 1에서, 질의 시퀀스와 ε-매치하는 서브시퀀스는 최소한 하나의 J-디스조인트 윈도우를 포함하여야 한다. 다시 말해서, 정리 1의 ρ가 1 이상이어야 한다. 그러면, 다음 과정에 의하여 질의 시퀀스 길이와 윈도우 크기와의 관계식을 구할 수 있다.

$$
\begin{aligned}
\rho = \lfloor (Len(Q)-b)/\omega \rfloor \geq 1 & \Leftrightarrow (Len(Q)-b)/\omega \geq 1 \\
& \Leftrightarrow Len(Q) \geq \omega + b \\
& \quad (1\text{이 정수이므로 } \Leftrightarrow \text{가 성립함})
\end{aligned}
$$

그러므로, 주어진 최소 질의 시퀀스 길이가 Min(Q)이면, $Len(Q) \geq \omega + b$에 의하여 $\omega \leq Min(Q) - b$이 성립한다. 그런데, 정리 1에서 $0 \leq b \leq J-1$이고, 모든 b에 대해 $\omega \leq Min(Q) - b$가 만족해야 하므로 $\omega \leq Min(Q) - J + 1$이 성립한다. 그리고, ω는 J의 배수로 한정하였으므로, 최대 윈도우 크기는 $\lfloor (Min(Q)-J+1)/J \rfloor * J$가 된다.

부록 D: 보조정리 7의 증명

데이타 시퀀스 S가 n개의 J-슬라이딩 윈도우로 나누어진다고 하자. 그러면, S에 포함된 마지막 J-슬라이딩 윈도우는 $s_n^J (= S[(n-1)*J+1:(n-1)*J+\omega])$이 되므로, $(n-1)*J+\omega \leq Len(S)$가 성립한다. 결국, $n \leq (Len(S)-\omega)/J+1$이 되고, n이 정수이므로 $n \leq \lfloor (Len(S)-\omega)/J \rfloor + 1$이 성립한다. 그런데, s_n^J이 마지막 윈도우이므로, $n = \lfloor (Len(S)-\omega)/J \rfloor + 1$이 된다. 그 이유는 만일 $n < \lfloor (Len(S)-\omega)/J \rfloor + 1$이라면, n과 $\lfloor (Len(S)-\omega)/J \rfloor + 1$이 정수이므로 $(n+1) \leq \lfloor (Len(S)-\omega)/J \rfloor + 1$이 성립하고, 결국 s_n^J의 바로 다음 J-슬라이딩 윈도우인 s_{n+1}^J도 S에 포함되기 때문이다. 따라서, 색인에 저장해야 하는 점의 개수는 S의 J-슬라이딩 윈도우 개수인 $\lfloor (Len(S)-\omega)/J \rfloor + 1$이 된다.

문 양 세
정보과학회논문지 : 데이타베이스
제 28 권 제 1 호 참조

한 욱 신
1994년 2월 경북대학교 컴퓨터공학과 학사. 1996년 2월 한국과학기술원 전산학과 석사. 1996년 3월 ~ 현재 한국과학기술원 전산학과 박사과정. 관심분야는 객체지향 DBMS, 객체 관계형 DBMS, 객체 캐쉬 관리 및 선인출

황 규 영
정보과학회논문지 : 데이타베이스
제 28 권 제 1 호 참조