# 클러스터 세그먼트 인덱스를 이용한 단백질 이차 구조의 효율적인 유사 검색

## (Clustered Segment Index for Efficient Approximate Searching on the Secondary Structure of Protein Sequences)

서 민 구 † 박 상 현 †† 원 정 임 †††
(Minkoo Seo) (SangHyun Park) (JungIm Won)

**요 약** 단백질 일차 구조(아미노산 배열)에 대한 상동 검색은 유전자나 단백질의 기능과 진화 과정을 유추하기 위한 필수 연산이다. 그러나 진화 단계가 멀리 떨어진 경우 단백질 일차 구조는 보존되지 않기 때문에 단백질의 공간적 구조에 대한 유사 검색을 통해서만 진화 단계를 유추할 수 있다. 따라서 본 논문에서는 단백질의 공간적 구조를 표현하는 단백질 이차 구조를 대상으로 하여 RDBMS상에 쉽게 구현이 가능한 인덱싱 방안을 제안한다. 제안된 인덱싱 방안은 클러스터링 기법과 LookAhead 개념을 활용하여 Exact Match, Range Match, Wildcard Match 질의를 신속하게 처리한다. 제안된 방법의 우수성을 검증하기 위하여 실제의 단백질 데이타를 대상으로 성능 평가를 수행하였다. 실험 결과에 의하면, 제안된 방법은 기존의 방법과 비교하여 Exact Match의 경우 6.3배까지, Range Match의 경우 3.3배까지, Wildcard Match의 경우 1.5배까지의 개선된 검색 성능을 가지는 것으로 나타났다.

**키워드** : 인덱싱 기법, 단백질 이차 구조, 유사 검색

***Abstract*** Homology searching on the primary structure (i.e., amino acid arrangement) of protein sequences is an essential part in predicting the functions and evolutionary histories of proteins. However, proteins distant in an evolutionary history do not conserve amino acid residue arrangements, while preserving their structures. Therefore, homology searching on proteins' secondary structure is quite important in finding out distant homology. In this manuscript, we propose an indexing scheme for efficient approximate searching on the secondary structure of protein sequences which can be easily implemented in RDBMS. Exploiting the concept of clustering and lookahead, the proposed indexing scheme processes three types of secondary structure queries (i.e., exact match, range match, and wildcard match) very quickly. To evaluate the performance of the proposed method, we conducted extensive experiments using a set of actual protein sequences. CSI was proved to be faster than the existing indexing methods up to 6.3 times in exact match, 3.3 times in range match, and 1.5 times in wildcard match, respectively.

**Key words** : Indexing method, Secondary structure of proteins, Approximate searching

## 1. 서 론

단백질 내 아미노산 서열이 단백질의 구조와 기능을 결정한다는 사실은 생물학자들에게 널리 알려져 있다. 따라서 새로이 발견된 단백질의 기능, 역할, 구조, 분류 등을 특성이 판명된 기존의 단백질 아미노산 서열과 비교함으로써 예측할 수 있다[1,2]

그러나 두 개의 단백질이 진화 단계에서 멀리 떨어져 있는 경우, 단백질의 구조가 유사하더라도 아미노산 서열은 서로 간에 크게 다를 수 있다[3-5]. 따라서 진화 단계에서 멀리 떨어진 단백질들 간의 상동 관계(homology) 분석에서는 아미노산 서열의 비교 보다는 구조 비교가 더욱 중요하게 다루어진다. 이러한 구조 검색 알고리즘 중에서 데이타베이스 접근과 연관하여 단백질 이차 구조의 비교를 통한 구조 비교 방법이 점점 더 많

· 이 논문은 2005년 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2005-206-D00015).
† 비 회 원 : 연세대학교 컴퓨터과학과
mkseo@cs.yonsei.ac.kr
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
sanghyun@cs.yonsei.ac.kr
††† 비 회 원 : 연세대학교 컴퓨터학과 교수
jiwon@cs.yonsei.ac.kr
논문접수 : 2004년 7월 22일
심사완료 : 2005년 12월 27일

은 주목을 받고 있다[3,6].

단백질의 이차 구조는 E (beta sheets), H (alpha helices), L (turns or loops)의 세 가지 타입의 구조를 나타내는 문자들로 구성된다. 이 때 각 문자는 산발적으로 나타나기 보다는 연속적으로 나타나는 경향이 있다[7,8]. 따라서 *'HHLLLLLEEE'* 의 구조가 관측될 확률이 *'HLLELLEELH'* 의 경우에 비해 높다.

이러한 특징에 근간하여 Hammel 등[8]은 세그먼트 기반의 인덱싱 기법을 제안하였다. 이 기법에서는, 동일한 타입 문자의 연속을 하나의 세그먼트로 묶고 세그먼트 내 Type, Len의 두 가지 속성에 대해 $B^+$ 트리를 구성하였다. 이 때, Type은 연속적으로 나타난 문자를 의미하며, Len은 문자가 연속되어 나타난 회수를 뜻한다. 예를 들어, *'HHLLLLLEEE'*는 *'HH/LLLLL/EEE'*의 3개 세그먼트로 분할되고, *(H, 2)(L, 5)(E, 3)*으로 표현된다.

세그먼트 방식은 단백질 이차 구조에 대한 효율적인 검색을 가능하게 하지만 다음과 같은 근본적인 한계를 가진다. 첫째, (Type, Len) 쌍의 분포가 균일하지 않다. 80,000개의 단백질에 대해 수행한 사전 실험 결과에 따르면, 87%의 E 세그먼트의 길이는 3에서 6 사이, 62%의 H 세그먼트의 길이는 5에서 14사이, 41%의 L 세그먼트의 길이는 3에서 6으로 조사되었다. 따라서 질의 내 모든 세그먼트가 이러한 *핫 스팟(hot spot)*에 해당한다면, 각 세그먼트에 대한 인덱스 검색은 수많은 중간 결과를 발생시켜 전체적인 검색 속도가 풀 테이블 스캔보다 더 나빠질 수 있다. 두 번째 더 중요한 한계는 가능한 (Type, Len) 쌍의 총 수가 좋은 정선도(selectivity)를 제공하기에 충분치 않다는 점이다. 80,000개 단백질에 대한 사전 실험 결과, 서로 다른 (Type, Len) 쌍의 총 수는 300개였으나, 인덱싱 대상이 되는 총 세그먼트의 개수는 3백만개 이상으로 조사되었다. 따라서 (Type, Len) 쌍 하나에 대해 검색되는 세그먼트의 개수는 평균 10,000개 이상이다.

본 논문에서는 클러스터 세그먼트 인덱스(CSI: Clustered Segment Index)를 사용한 단백질 이차 구조에 대한 효율적인 검색 기법을 제안한다. 제안된 기법은 *클러스터(cluster)*와 *룩어헤드(LookAhead)*를 사용하여 앞서 지적한 세그먼트 기반 인덱싱 기법의 한계를 극복한다. 사전에 정의된 개수만큼의 연속된 세그먼트는 하나의 클러스터로 묶여 CluStr, CluLen, CluLA의 총 3가지 속성으로 표현된다. 이 중 CluStr은 세그먼트의 Type 속성 값을 연결한 타입 문자열을 의미하며, CluLen은 세그먼트내 Len 값의 합을 의미한다. CluLA는 LookAhead로서, 현재 클러스터 뒤에 오는 세그먼트들의 Type 속성 값을 연결한 문자열을 의미한다. 클러스터를 표현하는 (CluStr, CluLen, CluLA)가 세그먼트를 표현하는 (Type, Len)에 비해 더 많은 경우의 수가 존재함은 분명하다. 따라서 클러스터를 사용한 인덱싱 및 질의 처리 기법을 사용하여 검색된 중간 결과를 줄이고 전체적인 질의 처리 성능을 향상시킬 수 있다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 상동관계 검색을 위한 시퀀스 검색 및 구조 검색에 관한 관련 연구를 기술한다. 3장과 4장에서는 제안된 인덱싱 기법과 질의 처리 알고리즘에 대해 각각 설명한다. 5장에서는 성능 평가를 통해 제안된 방법의 효율성을 보인다. 6장에서는 논문의 내용을 요약하고 결론을 내린다.

## 2. 관련 연구

BLAST[9,10]는 DNA와 단백질 시퀀스에서 가장 널리 사용되는 도구이다. BLAST는 기본적으로 순차 검색을 사용하나, 질의와 비교할 시퀀스의 개수를 줄이기 위한 휴리스틱 알고리즘을 사용한다. 그러나 BLAST는 두 가지 주요한 한계를 가진다[2]. (1) 고속 검색을 위해서는 전체 시퀀스를 메모리에 로드해야한다. (2) 순차 검색을 사용한 검색은 데이타베이스내 전체 시퀀스의 개수에 비례하는 수행시간이 필요하다. 이러한 문제점들로 인해, 유사 시퀀스 검색을 위한 인덱스 기반의 연구가 필요한 실정이다.

접미어 트리(suffix tree)[11]는 문자열 또는 시퀀스 검색을 위한 가장 좋은 방법 중 하나로 알려져 있으나, 공간 소모량이 매우 크다는 문제가 있다. 이러한 문제점을 해결하기 위하여, 메인 메모리 보다 더 큰 접미어 트리를 작성하기 위한 알고리즘[12]이 제안되었다. 그러나 접미어 트리의 내부 구조상 페이징을 적용하기 어려우며, 따라서 데이타베이스 시스템과의 연계가 어렵다[11,13].

RAMdb[14]는 단백질 일차 구조의 검색을 위한 인덱싱 시스템으로서, 실험을 통해 휴리스틱 기법에 비해 800배이상 빠름이 증명되었다. 그러나 사전에 정한 질의 길이보다 더 긴 질의가 주어질 경우 검색 성능이 크게 떨어지는 문제가 있다. 따라서 단백질 이차 구조 검색에 곧바로 적용하기 힘들다.

Hammel 등[11]은 세그먼트 기반의 인덱싱 기법을 제안하였다. 이 기법은 동일한 타입 문자의 연속을 하나의 세그먼트로 묶고 연속된 문자들의 타입과 길이에 $B^+$ 트리를 구성한다. 그러나 서론에서 언급한 바와 같이, 세그먼트 기반의 접근 방법은 좋은 정선도를 제공하지 않으며, 검색 성능에 있어 본질적인 한계가 있다.

VAST[15]와 DALI[16]는 3차원 구조 검색을 지원하는 알고리즘이다. VAST는 SSE(Secondary Structure Elements)의 개수가 $C_\alpha$, $C_\beta$ 원자의 개수보다 훨씬 적다

는 점에 착안하였다[4]. 따라서 VAST는 (1) SSE 쌍 배열의 고속 검출, (2) SSE의 그룹화, (3) 하부 구조 정렬에 대한 유사도 평가의 세 단계로 수행된다.

반면 DALI는 $C_\alpha$ 원자를 거리 행렬을 사용하여 비교한다. 각 단백질 마다 점 행렬(dot matrix)과 유사한 거리 행렬이 생성된다. 행렬내 각 점은 폴리펩타이드 체인을 따라 존재하는 $C_\alpha$ 원자 간의 거리와, 단백질 구조내 $C_\alpha$ 원자의 거리를 표현한다[4]. 따라서 DALI는 거리 행렬의 비교를 통해 주어진 질의와 구조가 유사한 단백질을 검색할 수 있다.

앞서 기술한 바와 같이, VAST와 DALI는 구조 검색의 문제를 다루며, 특히 이차 구조 요소의 삼차원 좌표 비교 문제를 해결한다. 제안하는 CSI는 이차 구조의 검색 문제 중 타입과 길이의 유사성을 비교하는데 중점을 두고 있어 이들과 다르다.

## 3. 세그먼트 테이블과 클러스터 세그먼트 테이블

본 논문에서 제안하는 CSI(Clustered Segment Index)는 [8]에서 기술된 세그먼트 테이블의 아이디어를 이용한다. 따라서 이 장에서는 세그먼트 테이블의 구조를 설명한 뒤, 제안된 인덱싱 기법에서 사용되는 주요 자료구조인 클러스터 테이블에 대해 설명한다.

### 3.1 세그먼트 테이블

세그먼트는 단백질 내에 존재하는 동일 타입 이차 구조의 연속적 출현으로 정의된다. 하나의 세그먼트는 연속되어 나타나는 문자를 표현하는 Type과 문자의 연속 회수를 표현하는 Len으로 표현된다. 예를들어, 시퀀스 $S_1$='*EEEHHLLEEE*'는 '*EEE/HH/LL/EEE*'의 세그먼트로 분할되어 *(E, 3)(H, 2)(L, 2)(E, 3)*으로 표현된다. 또한 세그먼트 내에는 각 세그먼트를 식별하는데 필요한 세그먼트의 출현 위치 정보가 추가적으로 저장된다. 예를 들어, $S_1$의 경우에는 0, 3, 5, 7이 각각의 세그먼트에 저장된다. 최종적으로 각 세그먼트의 정보는 세그먼트 테이블 내에 저장된다. 표 1은 $S_1$='*EEEHHLLEEE*'에 대한 세그먼트 테이블을 나타낸다.

표 1 $S_1$='*EEEHHLLEEE*'에 대한 세그먼트 테이블

| SegID | ProteinID | Loc | Type | Len |
|---|---|---|---|---|
| 1 | $S_1$ | 0 | *E* | 3 |
| 2 | $S_1$ | 3 | *H* | 2 |
| 3 | $S_1$ | 5 | *L* | 2 |
| 4 | $S_1$ | 7 | *E* | 3 |

### 3.2 클러스터 세그먼트 테이블

세그먼트 그 자체는 정선도에 있어 한계가 있다. 따라서 사전에 정의한 수만큼의 연속된 세그먼트를 하나의 클러스터로 묶고, 이 클러스터를 저장함으로써 단백질 이차구조에 대한 표현의 구분이 더 잘 이루어지게 할 수 있다. 클러스터 세그먼트 테이블의 생성은 다음과 같은 절차에 따라 이루어진다.

(1) 각 단백질 시퀀스 $S$를 세그먼트의 연속으로 표현한다. 이 때, $N_S$를 $S$로부터 생성된 세그먼트의 총 개수라 하자.

(2) $k \in [0, \min(\lfloor \log_2(N_S) \rfloor, MaxK)]$ 의 각 $k$에 대해 다음을 수행한다.

(a) $2^k$ 크기의 슬라이딩 윈도우를 사용하여, $2^k$ 의 연속된 세그먼트로 구성된 클러스터들의 집합을 구한다.

(b) 각 클러스터를 $CST_k$ 로 명명된 클러스터 테이블에 저장한다.

단계 1의 *MaxK* 는 생성되는 클러스터 테이블의 개수를 조정하는데 사용되는 시스템 파라미터를 의미한다. $i$번째 세그먼트 대해 $T_i$를 Type, $L_i$를 Len의 값이라 할 때, $(T_1, L_1)(T_2, L_2)...(T_2^k, L_2^k)$ 로 나열되어있는 세그먼트들을 고려해 보자. 그러면, 각 세그먼트의 Type 속성 값을 연결한 CluStr과, Len 속성 값들을 모두 더한 CluLen을 이용하여, 클러스터 $(CluStr=T_1 \cdot T_2 \cdot ... T_2^k, CluLen=L_1+L_2+...+L_2^k)$가 생성된다. 예를 들어, *(H, 2)(L, 2)*가 하나의 클러스터로 묶이면, 이 클러스터는 *(HL, 4)*로 표현된다. 그림 1은 표 1에 보인 시퀀스 $S_1$에 대한 클러스터 세그먼트 생성 과정을 표현하고 있다.

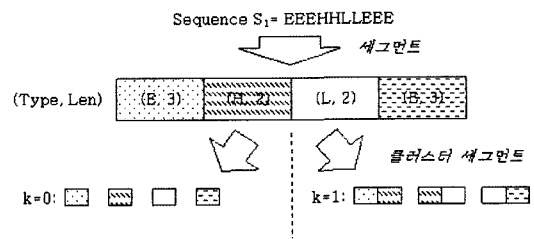


그림 1 $S_1$='*EEEHHLLEEE*' 에 대한 클러스터 세그먼트 생성

클러스터에는 그 뒤를 따르는 다수의 세그먼트가 있을 수 있다. 이들을 표현하기 위하여 이들 세그먼트의 Type 속성 값을 연결하여 LookAhead에 해당하는 CluLA를 생성한다. CluLA의 최대 길이는 저장 공간의 효율성을 고려한 시스템 파라미터인 *MaxCluLA*를 사용해 제한한다. 각 클러스터 세그먼트 테이블의 스키마는 표 2에 보인다.

예를 들어, *MaxK=1*, *MaxCluLA=2*일 때 $S_1$='*EEEHHLLEEE*'에 대한 클러스터 세그먼트 테이블은 표 3의 $CST_0$과 $CST_1$과 같이 작성된다.

표 2 클러스터 세그먼트 테이블의 스키마

| 필드 명 | 설 명 |
|---|---|
| ID | 단백질 식별자 |
| LoC | 클러스터의 시작 위치 |
| CluStr | 세그먼트의 Type 속성들을 연결하여 만든 타입 문자열 |
| CluLen | 세그먼트의 Len 속성들의 합으로 구한 클러스터의 길이 |
| CluLA | 현재 클러스터 뒤에 오는 세그먼트들의 Type 값을 연결하여 만든 타입 문자열 |

표 3 $S_1$='*EEEHHLLEEE*'에 대한 세그먼트 클러스터 테이블 $CST_0$, $CST_1$

$CST_0$

| ID | Loc | CluStr | CluLen | CluLA |
|---|---|---|---|---|
| $S_1$ | 0 | *E* | 3 | *HL* |
| $S_1$ | 3 | *H* | 2 | *LE* |
| $S_1$ | 5 | *L* | 2 | *E* |
| $S_1$ | 7 | *E* | 3 | |

$CST_1$

| ID | Loc | CluStr | CluLen | CluLA |
|---|---|---|---|---|
| $S_1$ | 0 | *EH* | 5 | *LE* |
| $S_1$ | 3 | *HL* | 4 | *E* |
| $S_1$ | 5 | *LE* | 5 | |

## 4. 질의처리

### 4.1 전체적인 질의처리 알고리즘

*MaxK+1*개의 클러스터 테이블 $CST_0$, $CST_1$, ..., $CST_{MaxK}$와 이들에 대한 $B^+$ 트리가 각각 작성되었다고 가정하자. 이 때, 이들 테이블과 인덱스를 사용한 전체적인 질의 처리는 다음과 같이 수행된다.

(1) 질의 $Q$를 세그먼트들의 연속으로 변환한다. 이 때, $N_S$를 $Q$로부터 구한 세그먼트들의 개수라 하자.

(2) $k = \min(\lfloor \log_2(N_Q) \rfloor, MaxK)$의 식을 사용해 대상 테이블 $CST_k$를 구한다.

(3) 세그먼트로 변환된 질의들을 $n(=\lceil N_Q/2^k \rceil)$개의 중첩되지 않는 서브 질의로 분할한다. 각각의 서브 질의는 $2^k$개의 세그먼트를 포함하게 되며, 서브 질의 중 최종 2개의 질의는 $N_Q$가 $2^k$의 배수가 아닐 경우 중첩될 수 있다. (그림 2 참조)

(4) 각 서브 질의 $q_i(i=1, 2, ..., n)$에 대해 다음을 수행한다.

(a) CluStr, CluLen, CluLA 값을 계산한다. qCluStr, qCluLen, qCluLA를 이 단계에서 계산된 이들 각각의 값이라 하자.

(b) $CST_k$ 테이블에서 CluStr, CluLen, CluLA의 값이 qCluStr, qCluLen, qCluLA에 일치하는 튜플들을 검색한다. 검색 시 사전에 작성된 $B^+$ 트리가 사용된다.

(5) $n$개의 중간 결과 집합에 대해 ID와 Loc을 조인 속성으로 하여 소트 머지 조인을 수행한다.

(6) 후처리를 통해 착오 채택(false alarm)을 검사 및 제거한다.

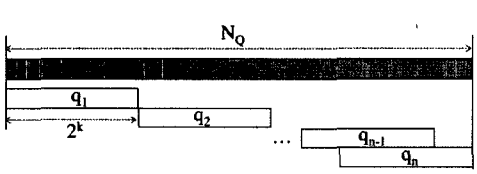


그림 2 질의 Q로부터 생성된 서브 질의

### 4.2 Exact Match 질의

Exact Match 질의는 $Q=<T_1(L_1)T_2(L_2)...T_{N_Q}(L_{N_Q})>$로 표현되며, 여기서 $T_i(\in\{E, H, L\})$는 $Q$내 $i$번째 세그먼트의 타입, $L_i$는 해당 세그먼트의 길이를 각각 의미한다. 대상 테이블인 $CST_k$가 이미 결정되었고, 질의 역시 각각 $2^k$의 세그먼트로부터 작성된 $n$개의 서브 질의로 분할되었다고 가정하자. 그러면 Exact Match 질의는 아래 Algorithm 1과 같이 처리된다.

서브질의 $q_i$의 처리 결과는 $N_i$에 저장된다. 만약 $N_i$내 튜플의 개수가 사전에 정의한 임계값 $\varepsilon$ 보다 작다면, 이미 결과가 충분히 걸러진 것으로 간주한다. 이러한 경우에는, 곧바로 머지 소트 단계(라인 6)로 넘어감으로써 남아있는 서브 질의 $q_{i+1}$, $q_{i+2}$, ..., $q_n$에 대한 처리를 생략한다.

### 4.3 Range Match 질의

Range Match 질의는 $Q=<T_1(Lb_1\ Ub_1)T_2(Lb_2\ Ub_2)...T_{N_Q}(Lb_{N_Q}\ Ub_{N_Q})>$로 표현되며 여기서 $Lb_i$와 $Ub_i$는 각각 질의 $Q$내 $i$번째 세그먼트의 최소와 최대 길이를 의미한다. 동등 연산자 술어를 사용하는 Exact Match의 경우와 달리, Range Match의 경우 검색 조건이 범위 술어로 주어진다. 즉, 서브 질의 $q_i$의 *CluLen*에 대한 검색 조건은 '*CluLen* between *qLb* and *qUb*'의 형태로 주어지며 *qLb*는 서브 질의 $q_i$내 세그먼트들의 *Len* 속성 값들의 합에 대한 하한, *qUb*는 상한을 의미한다. 만약 $q_i$가 질의의 처음 $2^k$개 세그먼트로 구성되었다면, *qLb*와 *qUb*는 각각 $qLb=Lb_1+Lb_2+...+Lb_2^k$, $qUb=Ub_1+Ub_2+...Ub_2^k$로 계산된다. 따라서 *qLb*와 *qUb*의 차가 클 경우, *CluLen*에 대한 검색 범위가 커져 $q_i$를 처리하는데 드는

Algorithm1: ProcessExactMatchQuery

```
입력: 질의 Q, 클러스터 세그먼트 테이블 CSTk, 임계값 ε
출력: 시퀀스 집합
1  for (each subquery qi from Q) do
2    Let qCluStr, qCluLen, and qCluLA be CluStr, CluLen, and CluLA of qi, respectively;
3    Ni := ExecuteQuery("select * from CSTk where CluStr=qCluStr
                         and CluLen = qCluLen and CluLA = qCluLA");
4    if (count(Ni) < ε) then break;
5  end for
6  Merge all Ni into N using ID and Loc as joining attributes;
7  answers := PostProcessing(N);
8  return answers;
```

비용 역시 커진다.

이러한 서브 질의 $q_i$의 처리 비용 증가 문제를 해결하기 위해, 본 논문에서는 선택적 클러스터링 기법(SCM: Selective Clustering Method)을 제안한다. SCM에서는 서브 질의 $q_i$를 이차 서브 질의의 집합으로 분할하고, 이들 중 검색 공간이 가장 작을 것으로 예상되는 이차 서브 질의를 질의 $q_i$ 대신 수행한다. 다시 말해, 서브 질의 $q_i$가 $2^k$의 세그먼트를 갖고 있을 때, 이차 서브 질의는 $2^{k'}$ ($k' \in \{0, k\}$)개의 세그먼트로부터 생성된다. 그림 3은 서브 질의 $q_i$에 대한 이차 서브 질의를 나타낸다.

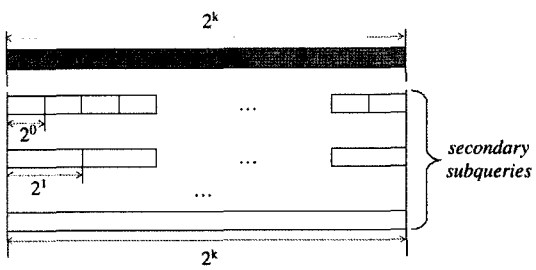


그림 3 서브질의 $q_i$에 대한 이차 서브 질의들

예를 들어 *MaxK=1*, *MaxCluLA=2*일 때, 질의 *Q=<E(3 5)H(3 6)L(3 7)>*을 고려해보자. 이 때, $k$는 $\min(\lfloor \log_2 3 \rfloor, 1) = 1$로 계산된다. 따라서 모든 서브 질의는 $2^1$개의 세그먼트로 구성되며, 서브 질의 $q_1$의 경우 $q_1$*=(qCluStr=EH, qCluLen=[3+3, 5+6], qCluLA=L)*로 표현된다.

$q_{i,j}$를 $i$번째 서브 질의에 대한 $j$번째 이차 서브 질의라 하자. 그러면 SCM에서는 세 개의 이차 서브 질의가 $q_1$으로부터 생성된다. *k'=0* 일 경우, 두개의 이차 서브 질의 $q_{1,1}$*=(qCluStr=E, qCluLen=[3, 5], qCluLA=HL)*, $q_{1,2}$*=(qCluStr=H, qCluLen=[3, 6], qCluLA=L)*가 생성되며 이들 각각은 내부에 $2^0$개의 세그먼트를 갖는다. 이와 마찬가지로, *k'=1*일 경우, 하나의 이차 서브 질의 $q_{1,3}$*=(qCluStr=EH, qCluLen=[6,11], qCluLA=L)*이 생성되며, 이 서브 질의는 $2^1$개의 세그먼트로 부터 구성된 것이다. 이들 이차 서브 질의 중, 실행 시 가장 적은 수의 튜플을 반환 할 것으로 예상되는 정선도 높은 이차 서브 질의를 선택하여 수행한다. 예를 들어, $q_{1,2}$가 그러한 이차 서브 질의로 예상되었다면 $q_1$대신 $q_{1,2}$가 실행되게 된다. $q_1$의 실행을 마치면 $q_2$*=(qCluStr=HL, qCluLen=[3+3, 6+7])* 역시 마찬가지로 처리된다.

SCM의 효율성은 이처럼 이차 서브 질의가 반환할 튜플의 수를 얼마나 효율적으로 표현하고 정확히 예측하는가에 달려있다. 정확도를 만족스러운 수준에서 유지하면서 필요한 정보를 효율적으로 표현하기 위해, 본 논문에서는 CluLen과 CluStr 각각에 대한 히스토그램을 별도로 유지한다. 표 4에 이들 두개 히스토그램에 대한 스키마를 보이며, 이차 서브 질의가 반환하는 튜플의 수를 예측하는 알고리즘은 Algorithm 2에 보인다.

### 4.4 Wildcard Match 질의

Wildcard Match 질의는 $Q=< T_1(Lb_1\ Ub_1)T_2(Lb_2\ Ub_2)...$

표 4 CluLen 및 CluStr에 대한 히스토그램

CluLen 히스토그램

| 필드 명 | 설 명 |
|---|---|
| hK | 클러스터의 k 값 |
| hCluLen | 클러스터의 CluLen 값 |
| #Clusters | k값이 hK, CluLen값이 hCluLen과 같은 클러스터의 총 개수 |

CluStr 히스토그램

| 필드 명 | 설 명 |
|---|---|
| hK | 클러스터의 k 값 |
| hCluStr | 클러스터의 CluStr 값 |
| #Clusters | h값이 hK, CluStr값이 hCluStr과 같은 클러스터의 총 개수 |

Algorithm2: Estimate_SizeOf-ResultSet

입력: 이차 서브 질의 $q_{i,j}$

출력: 결과 집합 내 튜플 개수의 예상 값

```
1 Let qCluStr be CluStr of q_{i,j};
2 Let [qLb, qUb] of q_{i,j} be the range of qCluLen;
3 Suppose that q_{i,j} is composed of 2^{qK} segments;
4 N1 := ExecuteQuery("select sum(#Clusters) from CluStrHistogram where hK=qK");
5 N2 := ExecuteQuery("select #Clusters from CluStrHistogram where hK=qK
   and hCluStr=qCluStr");
6 N3 := ExecuteQuery("select #Clusters from CluLenHistogram where hK=qK
   and hCluLen between qLb and qUb");
7 return N3 x N2 / N1;
```

$T_{N_Q}(Lb_{N_Q}\ Ub_{N_Q})$>로 표현되며 $T_i$는 {E, H, L, ?} 중 한 개의 값을 가질 수 있다. $Lb_i$와 $Ub_i$는 Range Match의 경우와 동일한 의미를 갖는다. 단, $T_i$ 는 "?"를 사용해 $i$ 번째 세그먼트가 임의의 이차 구조임을 지정할 수 있는 차이가 있다. 이와 같은 임의의 타입을 처리하기 위하여 Wildcard Match의 경우, Algorithm 1의 'CluStr= qCluStr' (라인 3)을 'CluStr like qCluStr'로 수정하여 사용한다.

Wildcard Match 질의에서도 CluStr과 CluLen의 검색 범위가 모두 확장될 수 있으므로, SCM을 적용한다. 이때, qCluStr이 "?"를 포함할 수 있으므로 Algorithm 2의 'hCluStr=qCluStr' (라인 5) 대신 'hCluStr like qCluStr'을 사용하도록 한다.

## 5. 성능 평가

### 5.1 실험 환경

실험에서는 512MB의 메인 메모리와 7200rpm의 하드 디스크가 장착된 2대의 펜티엄 4 PC를 사용하였다. 한 대의 PC에는 상용 RDBMS인 Oracle 8i를 설치하여 단백질 데이타를 저장하였으며, 이 데이타를 사용하여 세그먼트 테이블, 클러스터 세그먼트 테이블, 세그먼트 기반 인덱싱 기법[8]의 인덱스, CSI를 구성하였다. 다른 한대의 PC에는 검색 알고리즘을 구현하였다.

단백질 이차 구조 정보는 PREDATOR[17,18]를 PIR [19]로 부터 다운로드한 단백질 아미노산 서열에 적용하여 생성하였다.

제안된 기법인 CSI의 효율성을 평가하기 위하여, 세그먼트 기반 검색 기법인 MISS(1), MISS(2), SSS와 성능을 비교하였다. MISS(n)은 질의로부터 가장 정선도가 높은 $n$ 개의 세그먼트를 뽑은 뒤 이들 각각을 서브 질의로 취급하고, $B^+$ 트리를 사용해 세그먼트 테이블에 대한 검색을 수행한다. SSS는 질의로부터 가장 정선도가 높은 세그먼트를 선택한 뒤, 세그먼트 테이블에 대한 풀 테이블 스캔을 수행한다.

### 5.2 파라미터 설정

시스템 파라미터인 *MaxK*와 *MaxCluLA*를 결정하기 위해, 80,000개 단백질 시퀀스를 사용한 사전 실험을 수행하였다. 앞서 설명한 바와 같이 *MaxK*는 클러스터 테이블의 개수, *MaxCluLA*는 LookAhead의 최대 길이를 지정하는데 사용된다.

5.2.1 *MaxK*

테이블에 저장된 전체 튜플 수에 대한 임의의 질의에서 검색된 튜플 수의 비로 정의되는 *정선도(selectivity)*를 *MaxK*의 최적 값 결정에 사용하였다. 그림 4는 클러스터 테이블 $CST_k$에 대한 (CluStr, CluLen)의 정선도를 나타낸다. 그림에서, $k$가 증가할수록 정선도가 점차 좋아지지만 $k$가 3을 넘으면 그 향상 정도가 크게 작아짐을 볼 수 있다. 따라서 이후 실험에서는 *MaxK*의 최적 값을 3으로 정하였다.

5.2.2 *MaxCluLA*

*MaxCluLA* 값이 커질수록 정선도는 점차 좋아진다. 이러한 정선도의 향상 정도(DIS: Degree of Improvement in Selectivity)를 평가하기 위하여, 다음의 식을 사용하여 DIS(%)를 구하였다.

$$\frac{selectivity\ of\ (CluStr,\ CluLen) - selectivity\ of\ (CluStr,\ CluLen,\ CluLA)}{selectivity\ of\ (CluStr,\ CluLen)} \times 100$$

그림 5는 *MaxCluLA* 값의 증가에 따른 DIS를 도시한 그림이다. 실험 결과에 따르면, 정선도의 향상 정도는 *MaxCluLA*가 커짐에 따라 점차 커졌으나 *MaxCluLA*값이 8이상이 되면 DIS의 증가 정도가 크게 작아졌다. 따라서 이후 실험에서는 *MaxCluLA*의 최적 값으로 8을 선택하였다.

### 5.3 CluStr, CluLen 히스토그램의 정확도

CluStr, CluLen 히스토그램을 사용한 이차 서브 질의가 반환하는 튜플 수의 예측에 대한 정확성을 평가하기 위하여 두개의 변수를 사용하였다. 첫 번째 변수인 *에러율(error rate)*은 $|N_a - N_p| / N_a$로 정의되며, 여기서 $N_a$

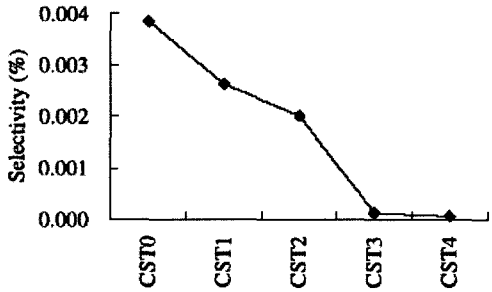


그림 4 각 $CST_k$ 테이블에서의 (CluStr, CluLen)의 정선도

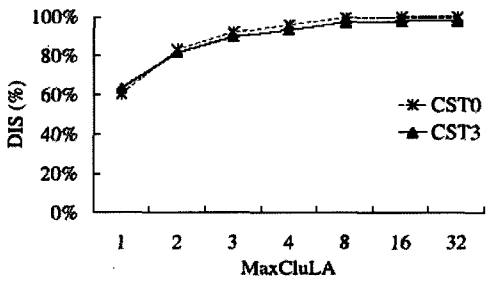


그림 5 MaxCluLA 증가에 따른 정선도 향상 정도 (DIS)

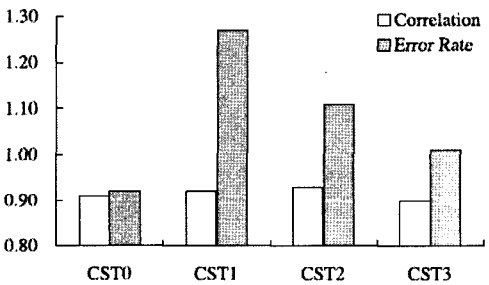


그림 6 각 $CST_k$ 테이블에 대한 제안된 예측 기법의 error rate와 correlation 평가 결과

는 이차 서브 질의가 반환한 튜플의 실제 개수, $N_p$는 CluStr, CluLen 히스토그램으로부터 구한 $N_a$의 예측 값이다. 두 번째 변수인 *상관계수(correlation)*는 $N_a$와 $N_p$의 선형적 관련성을 측정하는 값이다.

그림 6에서 볼 수 있듯이, error rate는 0.9에서 1.3 사이로 나타났다. 그러나 correlation은 최소 0.9 이상으로, $N_a$와 $N_p$가 높은 선형적 상관관계를 갖는 것으로 나타났다. 따라서 error rate의 값이 비록 작지 않으나, $N_a$ 값이 가장 작은 이차 서브 질의의 $N_p$ 값이 다른 이차 서브 질의의 $N_p$ 값에 비해 작을 확률이 매우 높음을 알 수 있다.

### 5.4 인덱스 크기

이 절에서는 CSI와 SSS, MISS(n)의 공간 요구량을 비교 평가한다. SSS는 세그먼트 테이블을 제외한 여타의 인덱스 구조를 갖지 않으므로, SSS의 공간 요구량은 세그먼트 테이블의 크기와 동일하다. 반면 MISS(n)은 세그먼트 테이블의 Type, Len 컬럼에 대한 $B^+$ 트리를 사용한다. 따라서 MISS(n)의 공간 요구량은 세그먼트 테이블의 크기와 $B^+$ 트리 인덱스 크기의 합으로 구할 수 있다. CSI는 클러스터 세그먼트 테이블과 함께 CluStr, CluLen 컬럼에 대해 작성된 $B^+$ 트리를 사용한다. 따라서 CSI의 공간 요구량은 세그먼트 테이블과 각 테이블에 대한 $B^+$ 트리의 합으로 구할 수 있다. 표 5에 보인 실험 결과에 따르면, 모든 기법의 공간 요구량이 단백질 개수에 비례하는 것을 알 수 있다. CSI는 다른 세가지 방법에 비해 수배의 공간을 필요로 하지만, 저장 공간에 대한 가격의 하락 및 검색 성능의 향상을 고려할 경우 이러한 공간 소모가 크게 문제되지 않을 것으로 판단된다.

표 5 SSS, MISS(n), CSI의 인덱스 크기 비교 (단위: KBytes)

| 단백질 시퀀스의 수 | SSS | MISS(n) | CSI |
|---|---|---|---|
| 20,000 | 20,288 | 35,626 | 149,640 |
| 40,000 | 37,160 | 65,258 | 272,350 |
| 60,000 | 51,992 | 91,304 | 378,907 |
| 80,000 | 67,688 | 118,852 | 481,516 |

### 5.5 인덱스 정선도

1장에서, 클러스터 세그먼트 테이블의 튜플들의 정선도가 세그먼트 테이블에 비해 좋음을 주장하였다. 이의 증명을 위해, 각 인덱싱 기법의 정선도를 이 절에서 비교하도록 한다. 클러스터 세그먼트 테이블 $CST_k$의 정선도는 $CST_k$에 저장된 전체 튜플 수에 대한 질의의 (CluStr, CluLen, CluLA)에 의해 검색된 튜플 수의 비로 정의된다. 세그먼트 테이블의 정선도는 테이블내 저장된 전체 튜플 수에 대한 질의의 (Type, Len)에 의해 검색된 튜플 수의 비로 정의된다. 그림 7에 보인 실험 결과에 따르면, $CST_k$의 정선도는 세그먼트 테이블에 비해 33배 이상 정선도가 좋은 것으로 나타났으며, 이러한 정선도의 향상에 따라 후처리를 통한 검증이 필요한 후보의 양을 크게 줄여준다.

### 5.6 질의 처리

5.6.1 질의 내 세그먼트 수 변화에 따른 질의 처리 시간 평가

질의 내 세그먼트 개수인 $N_Q$를 변화시키면서, CSI,

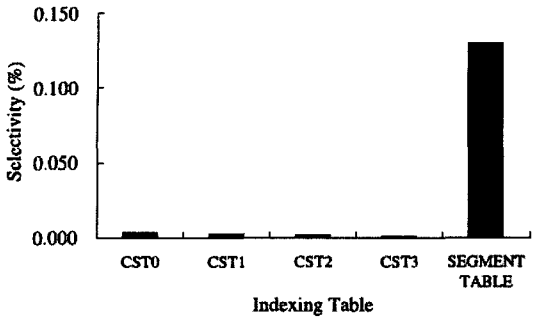


그림 7 $CST_k$ 테이블과 세그먼트 테이블의 정선도 비교

MISS(1), MISS(2), SSS의 질의 처리 시간을 측정하였다. 실험에서는 80,000개의 단백질 시퀀스가 사용되었으며 질의는 이들 시퀀스에서 랜덤 생성하여 사용하였다. 실험의 편의를 위해, Range Match와 Wildcard Match에서 질의 내 세그먼트들 중 중간 세그먼트의 길이에만 범위를 주었다. 이 때, 세그먼트 길이의 범위는 세그먼트 길이의 분포를 고려하여 30으로 정하였다. 마찬가지로, Wildcard Match 질의에서도 중간 세그먼트에만 와일드카드 문자 "?"를 사용하였다. 그림 8, 9, 10은 Exact Match, Range Match, Wildcard Match 각각의 질의 처리 시간을 나타낸다.

실험 결과, $N_Q$가 증가함에 따라 모든 기법의 질의 처리 시간이 감소하였다. 이는 $N_Q$가 커짐에 따라 질의 내에 정선도가 높은 세그먼트가 포함될 확률이 높아지기 때문이다. $N_Q$가 클 경우, CSI는 검색에 사용할 클러스터 테이블 $CST_k$를 결정함에 있어 더 큰 $k$를 사용할 수 있으므로 추가적인 성능 향상이 이루어진다. 따라서 CSI는 각 질의 유형에 대하여 가장 빠른 검색 성능을 보인 기법에 비해, Exact Match의 경우 1.7~13.0배, Range Match의 경우 1.3~6.0배, Wildcard Match의 경우 1.0~3.4배 빠르게 수행되었다.

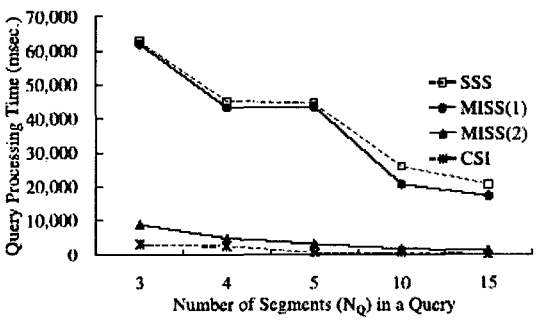


그림 8 $N_Q$ 증가에 따른 Exact Match 질의 처리 시간

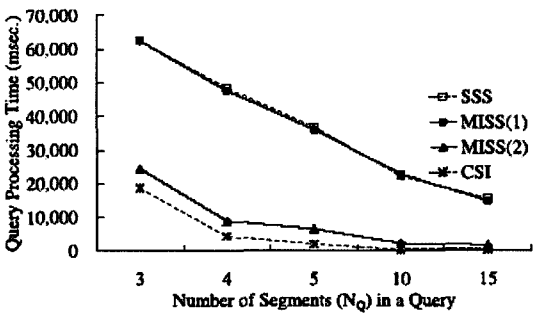


그림 9 $N_Q$ 증가에 따른 Range Match 질의 처리 시간

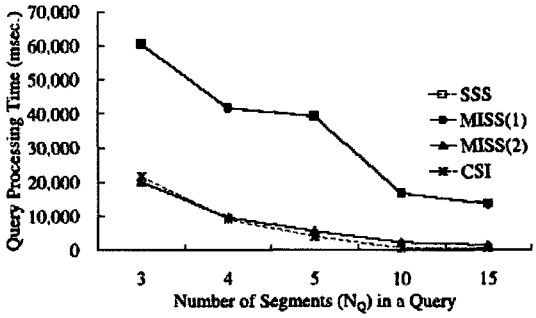


그림 10 $N_Q$ 증가에 따른 Wildcard Match 질의 처리 시간

5.6.2 데이타 셋 크기 변화에 따른 질의 처리 시간 평가

이 절에서는 단백질 시퀀스의 수 $N_{seq}$를 20,000에서 160,000으로 증가시켜가면서 CSI, MISS(1), MISS(2), SSS의 성능을 평가한 결과를 보인다. SSS와 MISS(1)은 앞 절의 실험에서 MISS(2)에 비해 대부분의 경우 느리게 수행되는 것으로 나타났으므로 본 실험에서는 제외하였다. 질의 내 세그먼트의 수는 5로 고정하였으며, Range Match 질의와 Wildcard Match 질의에서 세그먼트 길이의 범위는 세 번째 세그먼트에만 지정하였다. 그림 11, 12, 13에 보인 실험 결과에 따르면 CSI와 MISS(2) 모두 데이타 셋 크기에 따라 질의 처리 시간이 증가하는 것으로 나타났다. 또한 CSI는 Exact Match의 경우 4.3~6.3배, Range Match의 경우 3.0~

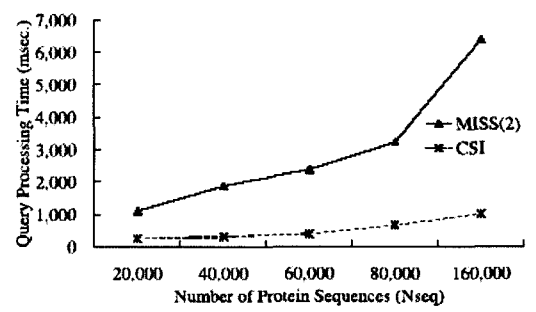


그림 11 $N_{seq}$ 증가에 따른 Exact Match 질의 처리 시간

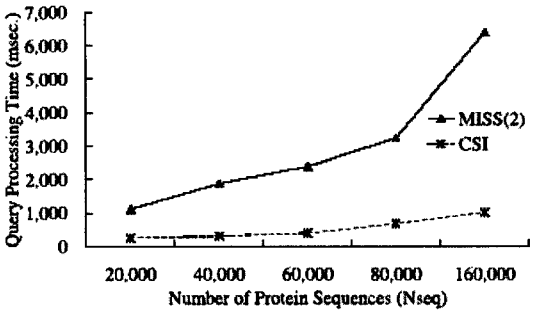


그림 12 $N_{seq}$ 증가에 따른 Range Match 질의 처리 시간

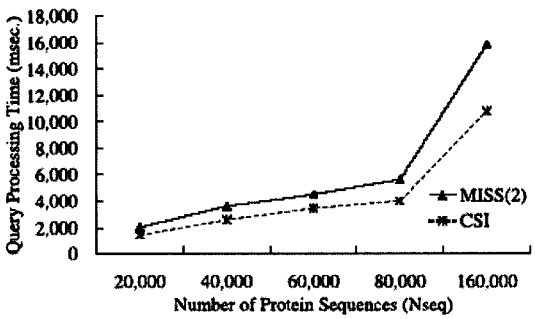


그림 13 $N_{seq}$ 증가에 따른 Wildcard Match 질의 처리 시간

3.3배, Wildcard Match의 경우 1.4~1.5배 MISS(2) 보다 빠르게 수행되었다.

## 6. 결 론

단백질 시퀀스의 아미노산 염기 서열이 아닌, 구조에 대한 유사 검색은 진화 단계에서 멀리 떨어져 있는 시퀀스를 검색하는데 있어 필수적인 연산이다. 따라서 본 논문에서는 단백질 이차 구조의 검색에 효율적인 인덱싱 기법인 CSI를 제안하였다. 제안된 기법은 클러스터링과 LookAhead를 사용하여 인덱싱되는 속성의 정선도를 높였다. 또한, Exact Match, Range Match, Wildcard Match를 위한 질의 처리 알고리즘을 제안 및 평가하였다. 실험 결과에 따르면, CSI는 MISS(2)에 비해 최대 Exact Match의 경우 6.3배, Range Match의 경우 3.3배, Wildcard Match의 경우 1.5배 빠르게 수행되는 것으로 나타났다.

## 참 고 문 헌

[1] B. Alberts, D. Bray, J. Lweis, M. Raff, K. Roberts, and J. D.Watson (3rd), *Molecular Biology of the Cell* (Garland Publishing Inc., 1994).

[9] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs, *Nucleic Acids Research* 25(17) (1997), pp. 3389-3402.

[10] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, Basic Local Alignment Search Tool, *Journal of Molecular Biology* (1990), pp. 403-410.

[7] Z. Aung, W. Fu, and K.-L. Tan, An Efficient Index-based Protein Structure Database Searching Method, *Proc. IEEE DASFAA Conf.* (2003), pp. 311-318.

[3] O. Camoglu, T. Kahveci, and A. K. Singh, Towards Index-based Similarity Search for Protein Structure Databases, *Proc. IEEE Computer Society Bioinformatics Conf.* (2003), pp. 148-158.

[14] C. Fondrat and P. Dessen, A Rapid Access Motif Database(RAMdb) with a Searching Algorithm for the Retrieval Patterns in Nucleic Acids or Protein Databanks, *Computer Applications in the Bioscience* 11(3) (1995), pp. 273-279.

[17] D. Frishman and P. Argos, Seventy-five Accuracy in Protein Secondary Structure Prediction, *Proteins* 27(3) (1997), pp. 329-335.

[18] D. Frishman and P. Argos, Incorporation of Long-Distance Interactions into a Secondary Structure Prediction Algorithm, *Protein Engineering* 9(2) (1996), pp. 133-142.

[15] J. F. Gibrat, T. Madel, and S. H. Bryant, Surprising Similarities in Structure Comparison, *Current Opinion in Structural Biology* 6(3) (1996), pp. 377-385.

[8] L. Hammel and J. M. Patel, Searching on the Secondary Structure of Protein Sequence, *Proc. VLDB Conf.* (2002), pp. 634-645.

[16] L. Holm and C. Sander, Protein Structure Comparison by Alignment of Distance Matrices, *Journal of Molecular Biology* 233(1) (1993), pp. 123-138.

[12] E. Hunt, M. P. Atkinson, and R.W. Irving, Database Indexing for Large DNA and Protein Sequence Collections, *The VLDB Journal* 11(3) (2002), pp. 256-271.

[6] P. Koehl, Protein Structure Similarities, *Current Opinion in Structural Biology* 11(3) (2001), pp. 348-353.

[4] D. W. Mount, *Bioinformatics* (Cold Spring Harbor Laboratory Press, 2000).

[5] A. Pastore and A. Lesk, Comparison of Globins and Physocyanins: Evidence for Evolutionary Relationship, *Proteins: Struct., Func., Gen.* 8(2) (1990), pp. 133-155.

[11] G. A. Stephen, *String Searching Algorithms* (World Scientific Publishing, 1994).

[13] H. Wang, C.-S. Perng, W. Fan, S. Park, and P. S. Yu, Indexing Weighted Sequences in Large Databases, *Proc. IEEE ICDE Conf.* (2003), pp. 63-74.

[2] H. E. Williams, Genomic Information Retrieval, *Proc. Australasian Database Conf.* (2003), pp. 27-35.

[19] C. H. Wu, L.-S. L. Yeh, H. Huang, L. Arminski, J. Castro-Alvear, Y. Chen, Z. Hu, P. Kourtesis, R. S. Ledley, B. E. Suzek, C. R. Vinayaka, J. Zhang, andW. C. Barker, The Protein Information Resource, *Nucleic Acids Research* 31(1) (2003), pp. 345-347.

서 민 구

2004년 2월 연세대학교 컴퓨터 과학과 졸업(공학사). 현재 카이스트 전산학과 석사과정 재학 중. 관심분야는 데이타베이스, 컴퓨터 보안, 바이오인포매틱스

박 상 현

1989년 2월 서울대학교 컴퓨터공학과(학사). 1991년 2월 서울대학교 컴퓨터공학과(석사). 2001년 2월 UCLA대학교 전산학과(박사). 1991년 3월~1996년 8월 대우통신 연구원. 2001년 2월~2002년 6월 IBM T. J. Watson Research Center Post-Doctoral Fellow. 2002년 8월~2003년 8월 포항공과대학교 컴퓨터공학과 조교수. 2003년 9월~현재 연세대학교 컴퓨터과학과 조교수. 관심분야는 데이타베이스, 데이타 마이닝, 바이오인포매틱스, XML

원 정 임

1992년 2월 한림대학교 전자계산학과(학사). 1997년 8월 한림대학교 컴퓨터공학과(석사). 2003년 2월 한림대학교 컴퓨터공학과(박사). 2000년 3월~2004년 2월 한림대학교 교양교육부 강의전담. 2004년 3월~현재 연세대학교 컴퓨터과학과 연구교수. 관심분야는 데이타베이스 시스템, 데이타 마이닝, XML 응용, 바이오 정보공학, 데이타베이스 보안