(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2015년01월06일
(11) 등록번호 10-1479735
(24) 등록일자 2014년12월30일

(51) 국제특허분류(Int. Cl.)
G06F 19/10 (2011.01) *C12Q 1/68* (2006.01)
(21) 출원번호 10-2012-0095457
(22) 출원일자 2012년08월30일
심사청구일자 2013년07월31일
(65) 공개번호 10-2014-0029788
(43) 공개일자 2014년03월11일
(56) 선행기술조사문헌
KR1020050059362 A
KR1020040051748 A
KR1020090118334 A
전체 청구항 수 : 총 4 항

(73) 특허권자
한국생명공학연구원
대전광역시 유성구 과학로 125 (어은동)
(72) 발명자
김태경
대전광역시 유성구 과학로 125
김경모
대전광역시 유성구 과학로 125
(뒷면에 계속)
(74) 대리인
이원희

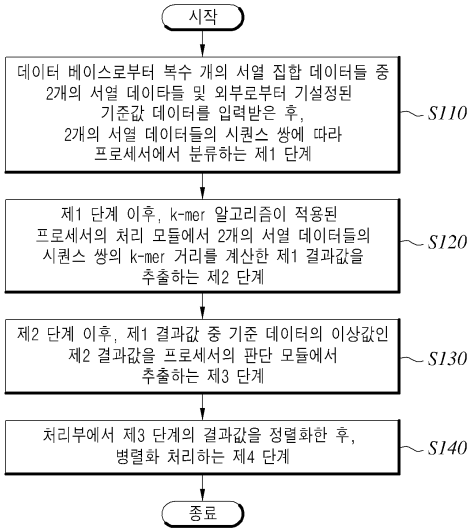
심사관 : 조현경

(54) 발명의 명칭 **FGA 알고리즘을 이용하는 서열 유사도 측정 시스템 및 이를 이용한 서열 유사도 측정 방법**

(57) 요 약

FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템 및 이를 이용한 서열(sequence) 유사도 측정 방법을 개시한다. 상기 FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템을 이용한 서열(sequence) 유사도 측정 방법은 데이터 베이스로부터 복수 개의 서열(sequence) 집합 데이터들 중 2개의 서열(sequence) 데이터들 및 외부로부터 기설정된 기준값 데이터를 입력받은 후, 상기 2개의 서열(sequence) 데이터들의 서열(sequence) 쌍에 따라 프로세서에서 분류하는 제1 단계; 상기 제1 단계 이후, k-mer 알고리즘이 적용된 상기 프로세서에서 상기 2개의 서열(sequence) 데이터들의 서열(sequence) 쌍의 k-mer 거리를 계산한 제1 결과값을 추출하는 제2 단계; 상기 제2 단계 이후, 상기 제1 결과갑 중 상기 기준 데이터의 이상인 제2 결과값을 상기 프로세서에서 추출하는 제3 단계; 및 처리부에서 상기 제3 단계의 결과값을 정렬화하여, 병렬화 처리하는 제4 단계를 포함한다.

대 표 도 - 도2



(72) 발명자

허보경

대전광역시 유성구 과학로 125

이상혁

대전광역시 유성구 과학로 125

이 발명을 지원한 국가연구개발사업

과제고유번호	KGM2111211
부처명	교육과학기술부
연구관리전문기관	기초기술연구회
연구사업명	주요사업(연구개발과제)
연구과제명	인터지노믹스 기반 생체방어연구
기 여 율	1/1
주관기관	한국생명공학연구원
연구기간	2012.01.01 ~ 2012.12.31

특허청구의 범위

청구항 1

복수 개의 서열(sequence) 집합 데이터들이 저장된 데이터 베이스;

상기 데이터 베이스로부터 상기 복수 개의 서열(sequence) 집합 데이터들 중 2개의 서열(sequence) 데이터들 및 외부로부터 기설정된 기준값 데이터를 입력받아 상기 2개의 서열(sequence) 데이터들의 유사도 값이 상기 기준값 이상인 데이터 값을 추출하는 복수 개의 프로세서들;

상기 복수 개의 프로세서로들 각각으로부터 출력된 결과값들을 정렬한 후, 병렬화 처리하는 처리부를 포함하며,

상기 복수 개의 프로세서들 각각은,

상기 2개의 서열(sequence) 데이터들의 k-mer 프로파일과 서열(sequence)의 길이 및 k-mer 크기에 따른 값을 이용하여 k-mer 거리를 산출하는 k-mer 알고리즘이 프로그래밍되어, 상기 2개의 서열(sequence) 데이터들의 k-mer 거리값인 제1 결과값을 출력하는 처리 모듈; 및

상기 제1 결과값 및 상기 기준값 데이터를 수신하여 상기 제1 결과값 중 상기 기준값 이상의 데이터인 제2 결과값을 출력하는 판단 모듈;을 포함하는 FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템.

청구항 2

데이터 베이스로부터 복수 개의 서열(sequence) 집합 데이터들 중 2개의 서열(sequence) 데이터들 및 외부로부터 기설정된 기준값 데이터를 입력받은 후, 상기 2개의 서열(sequence) 데이터들의 서열(sequence) 쌍에 따라 프로세서에서 분류하는 제1 단계;

상기 제1 단계 이후, 상기 2개의 서열(sequence) 데이터들의 k-mer 프로파일과 서열(sequence)의 길이 및 k-mer 크기에 따른 값을 이용하여 k-mer 거리를 산출하는 k-mer 알고리즘이 적용된 상기 프로세서의 처리 모듈에서 상기 2개의 서열(sequence) 데이터들의 서열(sequence) 쌍의 k-mer 거리를 계산한 제1 결과값을 추출하는 제2 단계;

상기 제2 단계 이후, 상기 제1 결과값 중 상기 기준 데이터의 이상값인 제2 결과값을 상기 프로세서의 판단 모듈에서 추출하는 제3 단계; 및

처리부에서 상기 제3 단계의 결과값을 정렬화한 후, 병렬화 처리하는 제4 단계를 포함하는 FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템을 이용한 서열(sequence) 유사도 측정 방법.

청구항 3

제2항에 있어서,

상기 k-mer 알고리즘은,

상기 2개의 서열(sequence) 데이터들의 k-mer 프로파일과 서열(sequence)의 길이 및 k-mer 크기에 따른 값을 아래에 기재된 식에 적용하여 k-mer 거리를 도출하는 것을 특징으로 하는 FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템을 이용한 서열(sequence) 유사도 측정 방법:

[식]

$$d_{X,Y} = \frac{\sum_{\tau} \min(n_X(\tau), n_Y(\tau))}{\min(\ell_X, \ell_Y) - k + 1}$$

여기서, X, Y는 서열(sequence) 데이터를 의미하고, $d_{X,Y}$는 서열 X 및 서열 Y 사이의 k-mer 거리(distance)를 의미하며, τ는 k-mer를 의미하고, nx(τ)는 서열 X에서의 해당 k-mer가 존재하는 횟수를 의미하며, n_Y(τ)는 서열 Y에서의 해당 k-mer가 존재하는 횟수를 의미하고, lx는 서열 X의 전체길이를 의미하며, l_Y는 서열 Y의 전체길이를 의미하고, k는 k-mer의 길이를 나타내는 상수값을 의미한다.

청구항 4

제2항에 있어서,

상기 제3 단계는,

상기 두 서열(sequence) 간의 k-mer 거리 및 글로벌 정렬 유사도에서 추출된 유전적 거리 간의 상관관계를 도출하는 단계를 포함하는 것을 특징으로 하는 FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템을 이용한 서열(sequence) 유사도 측정 방법.

명 세 서

기 술 분 야

[0001] 본 발명은 유사도 측정 시스템 및 이를 이용한 방법에 관한 것으로, 보다 상세하게는 두 개의 서열(sequence) 그룹에 대해 k-mer distance를 기준으로 글로벌 정렬 대상을 줄이고 멀티코어 서버 환경에서 병렬화를 통해 성능을 획기적으로 개선할 수 있는 FGA 알고리즘을 이용하는 서열 유사도 측정 시스템 및 이를 이용한 서열 유사도 측정 방법에 관한 것이다.

배 경 기 술

[0002] 서열(sequence) 정렬(Alignment)는 서열(sequence) 간의 상관관계를 보여주기 위해, 특히 상동성을 나타내기 위해 핵산이나 단백질의 서열(sequence)를 정렬(Alignment)하는 것을 말한다. 이것은 몇 개의 서열(sequence)를 정렬(Alignment)하는가에 따라, 정렬(Alignment)의 목표인 상동성의 종류에 따라 다음과 같이 구분한다.

[0003] 정렬(Alignment)하는 서열(sequence)의 갯수에 따라, 쌍 정렬(Alignment) - 2개의 서열(sequence)에 대한 정렬(Alignment) 멀티정렬(Alignment)-3개 이상의 서열(sequence)에 대한 정렬(Alignment) 상동성의 종류에 따라, 글로벌 정렬(Alignment) - 글로벌 상동관계에 대한 정렬(Alignment) 로컬 정렬(Alignment) - 로컬 상동관계에 대한 정렬(Alignment) 여기서, 정렬(Alignment)는, 관심 대상인 하나의 서열(sequence)와 상동성이 높은 서열(sequence)들을 알아내어 서열(sequence)의 기능을 유추하거나, 관련 있는 서열(sequence)들간의 정량적인 상관관계(진화적인 연관성과 같은)나 관련 기능 부위 등을 예측하기 위한 목적으로 이용된다.

[0004] 쌍 정렬(Alignment)는 상동성 검색을 위한 모든 문제의 기본이 되는 가장 기초적인 문제이므로, 보다 복잡한 문제의 알고리즘들은 대부분 쌍 정렬(Alignment) 알고리즘과 연관되어 있다.

[0005] 정렬(Alignment)할 서열(sequence) pair는 핵산-핵산, 또는 단백질-단백질과 같이 같은 종류의 서열(sequence)이며, 이는 멀티 정렬(multi-Alignment)의 경우도 마찬가지이다. 두 서열(sequence)를 비교하는 경우, 서로 동일한 종류의 핵산이나 단백질 서열(sequence)이어서 비교하는 서열(sequence)를 전체적으로 비교하여 최대의 상동성이 나타나도록 정렬(Alignment)하는 경우, 그러한 상동성을 글로벌 상동관계라고 하며, 이러한 정렬(Alignment)를 글로벌 정렬(Alignment)라고 한다.

[0006] 반면, 두 서열(sequence)의 어떤 부분 서열(sequence)가 높은 상동성을 가지는가를 나타내기 위해 정렬(Alignment)하는 경우, 그러한 상동성을 로컬 상동관계라고 하고, 그 정렬(Alignment)를 로컬 정렬(Alignment)라고 한다.

[0007] 높은 글로벌 상동관계를 나타내는, 같은 기능을 가진 단백질이나 핵산의 경우, 로컬 상동관계는 글로벌 상동관계를 보이는 부분과 유사한 부분을 나타내게 된다. 즉, 글로벌 정렬(Alignment)와 로컬 정렬(Alignment)가 유사한 결과를 보여주게 되는 것이다. 한편, 기능적으로 일부분만이 관련되어 있어서, 진화적으로도 일부분만이 상

동성을 보여줄 수 있을 경우는, 로컬 정렬(Alignment)를 실행하는 것이 효과적일 것이다.

[0008] 또한 유사한 기능의 서열(sequence)이나 글로벌 상동관계가 낮아서 쉽게 상동성을 나타내지 못하지만, 일부분에서는 높은 상동성을 보이는, 즉, 일부분에서 높은 로컬 상동관계가 있는 경우도 로컬 정렬(Alignment)를 실행하는 것이 효과적이다. 따라서, 많은 경우, 로컬 정렬(Alignment)가 상동성 서열(sequence)를 검색하기 위한 용도로 사용된다.

[0009] 서열(sequence) 정렬(Alignment)의 파라메터

[0010] 서열(sequence) 정렬(Alignment)에서 공통적으로 관련되는 주요 파라메터는 scoring matrix와 gap이다. 두 서열(sequence) 간의 상동성은, 정렬(Alignment)에서 연결된 모든 염기쌍, 혹은 아미노산쌍의 상동성의 합으로 계산되는데, 각 염기쌍과 아미노산쌍에 대해 상동성 값을 미리 정해둔 것이 바로 scoring matrix이다.

[0011] 따라서, 각 정렬(Alignment)를 구하는 과정에서 scoring matrix는 정렬(Alignment)의 형태와 선택에 있어 결정적인 역할을 하게 된다. 염기쌍에 대한 상동성은 문자적인 의미의 상동성만이(예로 1, 0과 같이) 일반적으로 사용되며, 일반적으로 scoring matrix 문제는 단백질에 대한, 즉 아미노산쌍에 대한 값을 나타낸다. 정렬(Alignment)에서 gap 발생에 대한 penalty를 얼마의 값으로 두느냐에 따라 gap 발생 장소와 길이, 횟수가 결정되어 정렬(Alignment)의 형태에 역시 중요한 영향을 주게 된다. 따라서, 현재의 정렬(Alignment) 알고리즘과 프로그램들의 지능화 수준을 고려할 때, 이들 파라메터의 의미를 파악하고 이에 맞추어 적용하는 것이 보다 의미있는 정렬(Alignment)를 구하기 위한 시작인 것이다.

[0012] 한편, 1970년 Needleman &Wunsch의 동적 프로그래밍 기법에 의한 쌍 정렬(Alignment) algorithm이 발표된 이후, Smith &Waterman 등이 일반적인 길이의 gap에 대해 이 algorithm을 확장함으로써, 두 서열(sequence) 간의 정렬(Alignment)를 위한 실용 가능한 프로그램들이 구현되기 시작하였다.

[0013] 이들 연구에 기반하여, 전산학의 그래프 알고리즘(graph algorithm) 이론이나, 패턴 매칭 알고리즘(패턴 matching algorithm) 등이 주로 생물학 패턴의 검색에 많이 응용되어 여러 응용 알고리즘들이 개발되어 왔으며, 데이터베이스의 개발과 함께 대량의 데이터를 대상으로 하여 빠른 실행 시간이 요구되어 여러 발견적 알고리즘(heuristic algorithm)들이 개발되었다.

[0014] Lipman, Wilbur, Pearson 등이 hashing과 window 설정의 방법을 사용하여 고안한 FASTP/FASTN(21,31,32)는 로컬 상동관계 정렬(Alignment)를 포함한 FASTA로 발전하여 데이터베이스 대상의 상동관계 search에 사용되었고, Altschul과 Karlin이 homologous candidate fragment 선정과 extension 방법을 사용하여 개발한 BLAST(3,18,19)는 데이터베이스 대상의 로컬 상동관계 search를 위해 가장 많이 활용되어 왔으며, 최근에는 BLAST의 결정적인 단점으로 gap 처리가 안되던 것을 해결한 BLAST2가 발표되어 BLAST를 대체하고 있다. 같은 목적으로 MPP 기종에서 병렬 프로그래밍된 MPsrch가 상용 프로그램으로 개발되어 있다.

[0015] [서열(sequence) 알고리즘의 종류]

[0016] 1. 동적 프로그래밍 알고리즘

[0017] 동적 프로그래밍은 전산학의 알고리즘 중의 한 유형으로, 글로벌 정렬(Alignment)로서의 쌍 정렬(Alignment)에 대한 최적화(optimal) 알고리즘으로 사용된다. 단백질 서열(sequence)의 한 예를 보자. ACDDDEFGR과 ACDEFHR의 두 서열(sequence)를 정렬(Alignment)하면, 다음과 같이 될 것이다.ACDDDEFGR ACDDDEFGR ACDDDEFGR||| || | | 또는, || | || | 또는, || ||| | 등등 ACD--EFHR AC-D-EFHR AC--DEFHR 이러한 정렬(Alignment)를 정량적으로 얻기 위해 몇 가지 가정이나 조건이 필요하다.

[0018] 정렬(Alignment)에서 서로 같은 자리로 연결된 각 아미노산쌍에 대해 어떤 값을 줄 것인가 하는 것이 첫 번 째 고려 사항이다. 간단히, 같은 아미노산쌍에게는 1을, 다른 경우에는 0을 준다고 하자. 또 하나는 위의 정렬(Alignment)에서 '-'로 표시된 gap이 발생하는 경우의 처리이다.

[0019] 즉. gap에 대해서는 어떤 값을 줄 것인가 하는 것이다. Gap의 생물학적 의미는 하나의 서열(sequence)에서

insertion이 발생했거나, 혹은 다른 서열(sequence)에서 deletion이 발생했다는 것이다. 실험실에서의 경험적인 통계와 계통관계 연구로부터 이러한 insertion/deletion은 다른 아미노산(핵산에서는 염기)로의 치환보다는 드문 사건이라고 일반적으로 생각할 수 있다. 따라서, 이것은 치환의 경우보다 상대적으로 낮은 값을 준다.

[0020] 엄격히 말하면, insertion과 deletion에 대해서는 다른 값을 주어야 하지만, 두 서열(sequence)를 비교할 경우, insertion과 deletion을 판정할 수가 없으며, 판정이 가능하다 해도(멀티정렬(Alignment)의 경우는 예측은 할 수 있다), 두 값에 대해 서로 상대적인 값을 줄 근거(계산방법)가 현재 불충분하다. 치환이 0으로 주어지면 이러한 gap에 대해서는 -2와 같이 임의의 더 작은 값을 설정할 수 있다. 이러한 조건에 의하면, 위의 세 정렬(Alignment)는 각각, 4,2,4의 값은 가지게 된다. 이러한 값을 상동성 값(상동관계 value)라고 하는데, 자연히 최대치를 가진 정렬(Alignment)를 선택하게 된다(첫 번째와 세 번째 정렬(Alignment)).

[0021] 먼저, 두 서열(sequence)의 각 자리를 행과 열로 하는 2차원 matrix가 만들어지는데, 이 matrix의 각 값은, 서열(sequence)의 시작 위치로부터 해당 행과열의 자리까지 두 서열(sequence)가 정렬(Alignment)되기 위한 최대의 상동관계 값을 나타내도록 한다. 이 값은 현재의 자리에 이를 수 있는 모든 정렬(Alignment)에서 현재 위치로 올 때의 각 경우의 최대치를 구하면 된다. 두 개의 서열(sequence) a1a2....am, b1b2....bn 이 각각 세로, 가로 방향으로 놓여 있을 때, ai와 bj가 구성하는 자리의 값 Hij는 다음과 같이 구한다. Hij = maxk,l(Hi-1k+gp, Hlj-1+gp, Hi-1j-1) + S(ai,bj) (단, gp=-2, 1 k j-1, 1 l i-1이고, S(a,b)=1 (a=b일 때), S(a,b)=0 (a b일 때)) 이렇게 구한 H의 마지막 행과 마지막 열 중에서 최대치를 구하면, 그것이 원하는 정렬(Alignment)를 이루는 최종 값이 되며, 이 값이 구성되는 경로를 구하면, 바로 정렬(Alignment)가 이루어진다.

[0022] 예의 matrix에서는 4가 최대치가 되며, 4를 이루는 경로는 따라가면, 다음의 정렬(Alignment)를 이루게 된다. ACDDDEFGR||| || | ACD--EFHR 가장 간단한 조건을 주었을 때의 글로벌 상동관계를 구하는 쌍 정렬(Alignment) 알고리즘은 위와 같은 동적 프로그래밍 알고리즘으로 구성된다.

[0023] 위 알고리즘은 Needleman &Wunsch와 Smith &Waterman에 의해 발표된 것으로 실제의 정렬(Alignment) program으로 실용화되기보다는 정렬(Alignment)를 위한 초기의 이론으로 사용되었다.

[0024] 실제 핵산이나 단백질 서열(sequence)에 대한 정렬(Alignment)를 할 때 고려할 몇 가지 점과 관련하여, 실용적인 알고리즘을 위해 추가해야 할 부분을 살펴보자. 아미노산쌍이나 염기쌍에 대한 연결 값을 나타내는 scoring matrix(혹은 substitution matrix)는 아미노산간의 진화적인 연관성이나, 기능적인 연관성을 반영하는 값을 사용한다. 위에서는 아미노산쌍에 대한 연결 값으로 1또는 0만을 사용했는데, 이것은 초기에 단순한 알고리즘 시험을 위해 사용되었던 UM(unitary matrix)이다.

[0025] 실제로는 PAM이나 BLOSUM 등이 사용된다. Gap 발생에 대해 주는 값인 gap penalty는 scoring matrix의 최저치보다 작은 값을 사용하는 것이 일반적이다. 위에서는 하나의 gap penalty만을 사용하였는데, 이는 진화적인 사건인 insertion / deletion은 하나의 gap에 대해 하나씩 연결된다고 단순화한 것이다. Gap의 길이에 따라 여러 형태의 gap penalty model을 설정할 수 있으나, 일반적으로는 affine gap cost를 사용한다. 위의 동적 프로그래밍 알고리즘은 서열(sequence) 길이를 L이라고 할 때, 각각 L2에 비례하는 실행시간과 memory를 요구한다.

[0026] 데이터베이스를 대상으로 상동성 검색을 하는 경우, NL2에 비례하는 실행시간이 필요하므로 용량이 큰 서버에서도 실행에 대한 부담이 매우 크다고 할 수 있다. 로컬 상동관계 검색에 대한 필요성과 이와 같은 실행 시간의 문제를 해결하기 위한 해결책으로 heuristic을 사용한 빠른 알고리즘들이 개발되어 BLAST, FASTA 등의 프로그램으로 사용되고 있다. Memory 사용에 대한 문제의 해결책으로는 linear space를 사용하는 정렬(Alignment) 알고리즘이 개발되었다.

[0027] 2. 상동성 검색 알고리즘

[0028] 기존에 알려진 서열(sequence)를 대상으로 하는 상동성 검색을 통해 연관성 있는 서열(sequence)의 전부나 일부분을 알아내는 것은 가장 생물학자들이 가장 빈번히 사용하는 분석이다. 앞에서 설명한 쌍 정렬(Alignment)를 그대로 적용하면 이러한 목적을 수행할 수 있으나, 실제로 실행 시간이 너무 길어서 실용적이지 못하다. 따라서, 위의 알고리즘의 취지를 살리되 빠른 알고리즘을 사용해야만 한다.

[0029] 이러한 시도로 최초로 발표되고 또 실용화된 것이 Lipman, Wilbur, Pearson 등에 의해 개발된 FASTP/FASTN이다. 이후 이 알고리즘의 민감도를 더욱 증가시키고, 로컬 정렬(Alignment) 알고리즘인 FASTA가 1988년 Pearson과 Lipman에 의해 발표되었다. 한편, 로컬 상동관계에 의한 데이터베이스 상동성 검색을 위한 새로운 알고리즘으로 BLAST가 1990년 Altschul 등에 의해 발표되었다. 최근까지 BLAST는 민감도와 속도 등의 면에서 가장 나은 해결책으로서 사용되고 있다. 한편에서는, MPsrch등과 같이 MPP 기종에서 실행되는 parallel programming에 의한 프로그램들도 개발되고 있으며, 이러한 프로그램들은 차츰 더 보편화될 것이다.

[0030] 3. FASTP/FASTN 알고리즘

[0031] 1983년 발표된 FASTP/FASTN 알고리즘의 핵심은 해싱(hashing)을 사용하여 질의 서열(sequence)와 상동성이 있는 부분을 sub-linear time에 검출하고, 가장 상동성이 높을 것으로 예측되는 부분에 대해서만 정렬(Alignment)를 시행함으로써, 실행 시간을 급격히 감소시킨 것이다. 이 알고리즘의 개요는 다음과 같다.

[0032] 1) 질의 서열(sequence)에 대해 k-mer(k tuple)를 해싱 함수를 사용하여 인덱싱하고, 해당되는 위치(즉, k-mer의 서열(sequence) 상의 위치)를 기록해 둔다.

[0033] 2) 데이터베이스상에서 비교할 서열(sequence)를 추출한다.

[0034] 3) 비교 서열(sequence)를 차례대로 scan하면서 k-mer가 질의 서열(sequence)에 있는지를 조사한다.

[0035] 4) 질의 서열(sequence)의 위치 i와 비교 서열(sequence)의 위치 j에서 k-mer가 일치할 경우, (i-j) diagonal에 일치된 것을 표시한다.

[0036] 5) 가장 많은 일치가 나타난 diagonal을 찾아내고, 이 diagonal의 주변 W만큼에 대해서만, score를 계산하고 최종 상동성 값을 구한다.

[0037] 6) 모든 데이터베이스 서열(sequence)에 대해 2)에서 5)를 반복한다.

[0038] 7) 최대 N개의 score를 보이는 서열(sequence)룰 순서대로 선택하고, 이들 각각을 질의 서열(sequence)와 동적 프로그래밍 알고리즘에 의해 정렬(Alignment)를 구한다.

[0039] 2)에서 4)까지의 과정은, gap 없이 정렬(Alignment)되었다고 가정할 경우 가장 많은 일치를 보이는 diagonal(동적 프로그래밍 알고리즘에서 Hij를 구하는 matrix에서)을 구하는 과정이다.

[0040] 과정 1)의 해싱 함수는 임의의 문자열 또는 검색키를 단시간에 접근하여 추출하기 위해 기억 장소 위치와 저장 데이터의 값을 연결하는 함수로 자료 저장 및 검색을 위해 사용되는 방법 중의 하나이다. 5)에서는 윈도우라 부르는 W만큼에 대해서만 Hij를 구하면 되므로, L2의 계산량 대신 LxW 만큼의 계산만을 필요로 한다. 이 부분은 heuristic으로 최대의 상동성을 보이는 정렬(Alignment)는 일반적으로 최대 일치 diagonal 주위에서 이루어진다고 가정한 것이다. 7)에서 각 서열(sequence)와 질의 서열(sequence)와의 잠정적인 최대치(heuristic에 의해 계산된)로 최종적으로 정렬(Alignment)할 서열(sequence)를 선택하고, 선택된 서열(sequence)와의 정렬(Alignment)는 optimal 쌍 정렬(Alignment)를 통해 구한다. 7)에서 optimal 정렬(Alignment)를 사용했지만, 이것은 최종적으로 선택된 서열(sequence)에 대해서만이고, 2)에서 5) 과정에서는 optimal 정렬(Alignment)에 비해 훨씬 빨리 최대치 정렬(Alignment)(heuristic에 의한)값을 구하므로 전체적으로 실행 시간은 급격히 감소하게 된다.

[0041] 4. FASTA 알고리즘

[0042] FASTA는 FASTP/FASTN에 비해 민감도가 증가되었고, 로컬 정렬(Alignment)이나 전체적에 걸친 로컬 정렬(Alignment)를 고려하므로 글로벌 정렬(Alignment)의 효과를 가진다(FASTA는 엄밀히 글로벌 정렬(Alignment)와 로컬 정렬(Alignment)의 중도적인 정렬(Alignment)를 수행하는 알고리즘이다). 알고리즘의 개요는 다음과 같다.

[0043] 1) 질의 서열(sequence)에 대해 hashing table (논문에서 lookup table)을 구성한다.

[0044] 2) 데이터베이스의 각 비교 서열(sequence)를 추출한다.

[0045] 3) 각 비교 서열(sequence)에 대해 k tuple이 일치하는 것을 조사한다.

[0046] 4) 최대 일치를 보이는 10개의 diagonal region을 구한다.

[0047] 5) 각 diagonal region에 대해 scoring matrix를 사용하여 비일치(mismatch)된 부분을 포함하여 score를 다시 계산하고, 각 region에 대해 maximal score를 나타내는 region을 구한다. (initial region)

[0048] 6) 특정 threshold 값 이상인 initial region 들로 구성할 수 있는 (joining penalty를 사용하여) 값을 구하고 이를 비교 서열(sequence)의 최종 선택에서 사용한다.

[0049] 7) 모든 데이터베이스 서열(sequence)에 대해 2)에서 6)를 반복한다.

[0050] 8) 최종적으로 구한 비교 서열(sequence) 각각에 대해 6)에서 연결된 initial region에 대해 FASTP와 같이 특정 윈도우 내에서 optimal 정렬(Alignment)를 구한다.

[0051] 3)의 과정은 FASTP/FASTN에서와 동일하다. 4)는 일치의 수와 일치된 자리간의 거리에 대한 공식에 의해 결정되며, FASTP/FASTN에서 1개의 diagonal을 선택한 것과는 달리 10개의 diagonal region을 선택한다. 5)에서 scoring matrix를 사용하고 또한, 일치되지 않은 부분도 scoring matrix를 사용하여 score에 포함시킴으로써, optimal 정렬(Alignment)에 의한 score에 근접한 값을 구할 가능성이 커지므로 결과적으로 민감도가 향상되는 것이다. 특히 한 diagonal이 아닌 여러 diagonal region을 joining하여 계산하므로, optimal 정렬(Alignment)에서의 정렬(Alignment)와 유사한 결과를 기대할 수 있으며, 따라서 민감도가 증가하게 된다.

[0052] 8)에서는 처음부터 optimal 정렬(Alignment)를 구하지 않고 윈도우 내에서만 정렬(Alignment)를 하지만, 사실상 optimal 정렬(Alignment)에 가까운 정렬(Alignment)를 기대할 수 있다. LFASTA는 이 알고리즘을 다소 수정하여 로컬 정렬(Alignment)만을 수행한다.

[0053] 5)에서 각 diagonal region 당 하나씩의 initial region을 구하는 대신(즉, 전체적으로 10개의 initial region을 구하는 대신), 특정 threshold 이상 되는 모든 region을 저장하고, 6)에서 region 간의 joining 대신, 각 initial region에 대한 로컬 정렬(Alignment)를 좌우를 확장해 가면서 찾는다.

[0054] FASTP/FASTN은 heuristic을 사용한 데이터베이스 상동성 검색으로서 로컬 상동관계를 찾는 듯이 보이지만 전체적으로 글로벌 상동관계를 계산하는 알고리즘이다.

[0055] 한편, FASTA는 FASTP/FASTN에 비해서 민감도가 향상되고, 또한 LFASTA를 통해 로컬 상동관계를 계산해 주기도 한다. 이들 프로그램들은 최근에도 나름대로 유용성을 가지고 여전히 사용되고 있다.

[0056] 5. BLAST

[0057] 1990년에 Karlin과 Altschul 등이 발표한 BLAST(Basic 로컬 정렬(Alignment) Search Tool)는 FASTA 알고리즘과 같이 heuristic을 사용하여 상동관계를 검색하기 위한 알고리즘으로 여러 가지 면에서 FASTA보다 진보된 방법이다. 알고리즘의 개요는 다음과 같다.

[0058] 1) 질의 서열(sequence)를 분석하여 질의 서열(sequence)의 <W-mer>와 score T 이상의 상동성을 가지는 <W-mer>를 모두 생성하여 이를 검색하기 위한 automata를 구성한다(Automata는 연관된 일련의 패턴을 효율적으로 검색하기 위해 일반적으로 사용될 수 있는 기법으로 이 경우 매우 적절한 활용으로 생각됨).

[0059] 2) 데이터베이스의 각 비교 서열(sequence)를 scan하면서 대해 1)에서 구성한 <W-mer>와 일치되는 것이 있는가를 automata를 따라 검사한다. 일치된 W-mer는 로컬 정렬(Alignment)를 위한 seed 서열(sequence)로 사용된다.

[0060] 3) 각 seed 서열(sequence)에 대해 질의 서열(sequence)와 비교 서열(sequence)의 좌우를 확장해 간다. 확장해 가는 도중 최대치 값보다 일정값 이상 차이가 나는 경우에 확장을 중단하고 이를 MSP(maximal segment pair)로 한다. 이 MSP가 특정 cutoff 값 S 이상이면, 이를 저장한다. BLAST는 로컬 정렬(Alignment) 만을 처리하며, 특히 gap이 없는 로컬 정렬(Alignment)만을 처리한다. 여기서, gap을 포함하도록 확장하면 속도가 급격히 떨어지고, 특히 gap을 포함했을 때의 유용성이 별로 없는 것으로 예측되므로 실제 프로그램에서는 gap을 처리하지 않는다.

[0061] 2)에서 W-mer 검색에서 핵산 서열(sequence)의 경우는 비교 서열(sequence)들을 4 base 당 하나의 byte로 압축

시켜 비교하기 때문에(BLAST 실행을 위한 서열(sequence) 데이터베이스가 핵산에 대해서는 압축파일로 구성되어 있다), scan 속도가 더욱 빨라진다. 민감도에 영향을 주는 주요 parameter는 W와 T 값이다.

[0062] 즉, 동일한 W에 대해 T가 작을 수록 automata를 이루는 W-mer의 list가 커지게 되고, 결과적으로 민감도는 증가하지만, 실행시간은 비례적으로 증가한다. Karlin 등은 데이터베이스가 특정 composition을 가질 경우에, 특정 score를 가진 서열(sequence) segment가 N개 나타날 확률을 구하는 방법을 공식화하였다(17). 따라서, BLAST에서 cutoff score S 이상을 가지는 MSP가 임의로 나타날 수 있는 기대치 E를 구할 수 있으며, 실제로 MSP가 N개 나타났을 경우, random하게 이 MSP가 나타날 확률을 구할 수 있다.

[0063] 따라서 BLAST 실행에서 S와 E는 동일한 의미를 나타내기 위해 사용되며, BLAST 결과에서 보여주는 확률 값은 검색된 MSP가 검색된 개수만큼 임의로 나타날 수 있는 확률이므로, 그 값이 작을수록 상동성의 중요도(significancy)가 있는 것으로 생각할 수 있다.BLAST는 여러 개의 프로그램으로 구현되었는데, 단백질 서열(sequence)를 PIR이나 SwissProt과 같은 단백질 데이터베이스에 대해 상동성 검색하는 blastp, 핵산 서열(sequence)를 GenBank와 같은 핵산 데이터베이스에 대해 상동성 검색하는 blastn, 단백질 서열(sequence)를 핵산 데이터베이스에 대해 상동성 검색하는(각 핵산 서열(sequence)를 6개의 reading frame으로 translation) tblastn, 핵산 서열(sequence)를 단백질 데이터베이스에 대해 상동성 검색하는(핵산 서열(sequence)를 6개의 reading frame으로 translation) blastx, 핵산 서열(sequence)를 핵산 데이터베이스에 대해 단백질로 변환하여 상동성 검색하는(각 핵산 서열(sequence)를 6개의 reading frame으로 translation) tblastx 등이 있다.

[0064] 6. BLAST2

[0065] BLAST2는 크게 gapped BLAST와 PSI-BLAST의 두 가지 프로그램으로 구성된다. BLAST에서 문제가 되어 왔고 BLAST 연구자들이 초기 논문에서 밝힌 것은 gap을 사용할 수 없고 gap을 처리하도록 확장하는 것이 별 의미가 없다는 것이었다. 그러나 BLAST의 새로운 version인 gapped BLAST에서는 알고리즘을 달리하여 gap을 처리할 수 있도록 하였고, 동시에 같은 정도의 민감도를 유지하면서도 실행 속도를 3배 가량 향상시켰다. 알고리즘에서 달라진 부분은 다음과 같다.

[0066] 변경 1) BLAST에서는 발견되는 W-mer를 seed로 사용하였다. gapped BLAST에서는 같은 diagonal에서 일정 distance 이내에 single hit들이 겹치지 않으면서 존재할 때, 이것을 extension을 시작할 candidate의 list에 둔다.

[0067] 변경 2) 특정 Sg 값 이상인 HSP에 대해 gapped extension을 실행한다. BLAST2의 근간이 되는 중요한 observation의 하나는, HSP를 이루는 segment는 같은 diagonal에 멀티ple hit을 가진다는 것이다. 즉, 발견된 W-mer로부터 최종적으로 HSP가 형성되는 경우에, 이 HSP는 일반적으로 또 다른 W-mer와도 관련이 되어 있다는 것이다. 바꾸어 말하면, 최소 2개의 hit을 가지지 않는 W-mer들은 HSP를 형성할 가능성이 거의 없으므로 seed로 선택할 필요가 없으며, 또한 두 개의 W-mer에 대해 동일한 HSP를 구하는 계산의 반복도 피할 수 있는 것이다. 이 방법은 seed의 selectivity를 향상시켜 전체적인 속도 향상에 크게 기여하는 부분이다.

[0068] 이 방법에서 BLAST와 같은 민감도를 유지하려면, T 값을 더 낮게 설정해야 하는데, 이것은W-mer의 개수를 증가시키게 된다.

[0069] 그러나, 반면, 2 hit을 가지는 W-mer만을 선택하게 됨으로써 최종적으로 extension에 포함되는 W-mer의 개수는 줄게 된다. 실제로 extension에 필요한 계산 시간이 W-mer 선정에 걸리는 시간보다 훨씬 많이 필요하므로, 이 방법은 실행 시간을 대폭 향상시키게 되는 것이다.

[0070] 변경 2)는 gap을 처리하기 위해 extension에서 gap을 고려하는 것이다. 그런데, 여기서 gapped extension은 이미 extension한 HSP 중 Sg 이상의 score를 가지는 것에 대해서만 실행하게 된다. Gapped extension은 upgapped extension에 비해 훨씬 많은 시간을 필요로 하나 Sg 에 의해 filter된 HSP의 개수는 그 중 적은 부분에 해당하므로 전체 실행 시간에 미치는 영향을 크지 않다. Gapped 정렬(Alignment)는 몇 개의 HSP를 연결시키게 되므로 같은 score에서 특정 HSP들을 놓칠 가능성을 줄이기 된다. 따라서, 실험적으로 T=13으로 설정해도 BLAST에서 T=11로 설정할 때에 해당하는 민감도가 나타나게 된다. 변경 1)에서는 T값이 반대로 설정되어야 하는데, 이 두 가지 효과를 합하면, 결국 같은 T를 설정해도 같은 정도의 민감도가 나타난다는 의미가 된다. 이와 같은 변경 조건에서는 실험적으로 gapped BLAST가 약 3배정도 실행 속도가 향상된 것으로 나타났다.

[0071] PSI-BLAST는(Position Specific Iterated BLAST), 각 iteration에서 상동성이 높은 서열(sequence)들을 추출하

여 이로부터 질의 서열(sequence)의 각 자리에 적합한 scoring matrix를 결정하여, 이 scoring matrix set을 다음 iteration에서 사용하는 것이다. 즉, 길이가 L인 질의 서열(sequence)에 대해 각 iteration마다 L x 20의 scoring matrix를 만드는 것이다. 이러한 방법은 약한 상동성을 보이는 서열(sequence)의 추출을 보다 sensitive하게 추출하기 위해 사용될 수 있다.

[0072] 즉, 동일한 서열(sequence) 쌍을 서로 다른 scoring matrix를 사용해서 상동성을 계산했을 때, 보다 진화적인 관계에 적합한 scoring matrix를 사용한 경우에 더 높은 상동성을 구할 수 있을 것이다. 이를 확장하여, 각 자리마다에 적합한(실제로 각 자리마다 임의의 서로 다른 치환율을 설정할 수 있다면 보다 정교한 상동성을 가지게 되므로) scoring matrix를 구하여 사용하면, 일반적인 scoring matrix로는 검출되지 않던 상동성이 검출될 가능성이 커질 것으로 기대할 수 있는 것이다. 실제로, 민감도 면에서 상동성 검색의 가장 중요한 문제는, long (evolutionary) distance, 혹은 weak similarity를 가지는 서열(sequence)에 대한 검출 문제이다. PSI-BLAST의 방법은 현재 시범적으로 시도되는 것이지만, 상동성 검색의 한계가 안고 있는 문제점을 극복하는 주요 방법 중의 하나로서, 매우 유용한 프로그램이 될 것이다.

[0073] 7. 상동성 중요도 판정 알고리즘

[0074] 여러 가지 정렬(Alignment)나 상동성 검색 프로그램의 결과에서 상동성 값은 여러 가지 형태로 나타난다. 지금까지 언급한 scoring matrix에 의한 score 값 자체, 정렬(Alignment) 상에서 동일하거나 유사한 것끼리의 비율을 나타내는 % 상동성 값 등이 일반적으로 사용되는 수치이다.

[0075] 그러나 이러한 절대치는 composition이나 질의 서열(sequence)의 길이, 로컬 정렬(Alignment)의 경우 검출된 서열(sequence) segment의 길이와 composition에 따라 해석이 달라질 수 있으므로, 이들 값을 통일된 기준치로 사용하기 어렵다.

[0076] 또한 이들 값은 확률적으로 random하게 나타날 수 있는 상동성에 비해 어느 정도 중요도가 있는 상동성(significant 상동관계)인지를 나타내 주지 못한다.FASTP/FASTN과 FASTA에서는 검색되어 정렬(Alignment)된 서열(sequence)에 대한 상동성의 중요도를 random 서열(sequence)와의 비교로 나타내는 별도의 프로그램으로 RDF, RDF2를 구현하였다.

[0077] 이들 프로그램은 정렬(Alignment)된 부분을 random 서열(sequence)들과 정렬(Alignment)했을 때의 score의 분포에 대해 평균과 표준 편차를 구하고, 각 정렬(Alignment)의 score가 그 분포 상에서 어디에 위치하는 가를 정규화 하여 나타냄으로써 그 중요성을 직관적으로 표시한다.

[0078] 한편, BLAST는 특정 composition과 score에 대한 통계 분석에 근거하여 score S를 가지는 N개의 MSP가 random하게 나타날 확률을 공식화하였으므로, 쉽게 이들 확률을 구하여, 결과 리스트에 이를 표시하므로, score와 더불어 이 확률을 참조함으로써 매우 직관적으로 상동성 중요도를 판정할 수 있다.

[0079] [서열(sequence) 정렬(Alignment)의 유전체 연구에의 응용]

[0080] 쌍 정렬(Alignment)는 유전자를 서로 비교하고자 할 때 가장 일반적인 도구가 된다. 유전체 연구가 급진전되어 데이터베이스의 크기가 수십 기가급에 이르고 있으므로, 데이터베이스를 대상으로 하는 상동성 검색에서는 실행 시간상 보다 효과적인 정렬(Alignment) 알고리즘이 점차로 필요하게 될 것이다.

[0081] 또한, 이미 EST에 대해 이루어지고 있는 작업처럼, 데이터베이스의 각 서열(sequence)를 GenBank나 PIR과 같은 주요 데이터베이스와 비교하는, 즉, 데이터베이스 대 데이터베이스의 상동성 검색과 같은 연구는, 오직 고성능의 컴퓨팅과 잘 조율된 서열(sequence) analysis 도구를 통해서만 수행이 될 수 있다. 한편, genome과 genome의 전체 서열(sequence) 분석과 같은 대용량의 작업도 기존 analysis 도구에 근거하여 특수하게 제작된 도구에 의해 이루어진다. Post-genome 프로젝트에 이르면, 이러한 분석은 방대하게 공개된 raw data인 서열(sequence) 데이터베이스로부터 기능 분석과 같은 복잡한 연구를 수행하는 주 연구 방법이 될 것이다.

[0082] 서열(sequence) 정렬(Alignment)는 ORF 예측, scoring matrix의 개량, 멀티정렬(Alignment)와 protein domain 분석, motif 생성, structural prediction 등의 거의 모든 서열(sequence) analysis에 있어 직접 적용되거나 하부 module로 연결되어 있다.

[0083] Post-genome 연구를 위해 서열(sequence) 정렬(Alignment)에서 개선을 필요로 하는 부분 또한 전반에 걸쳐

있다. 서열(sequence) 정렬(Alignment)는 서열(sequence) analysis의 여러 응용에서 가장 기본 module로 포함되어 있기 때문에 실행시간에서 중요한 역할을 하게 된다. 민감도를 높이기 위한 파라메터의 조절과 실행 시간의 단축이라는 두 가지가 상충되므로, 현재의 알고리즘으로 일정의 민감도를 유지하면서 대량으로 생산되고 있는 데이터의 계산을 처리하기는 매우 어렵다.

[0084] 간단한 예로, domain 분석을 위해 500개의 단백질에 대해 멀티정렬(Alignment)를 실행할 경우 50개의 단백질에 대한 경우보다 약 100배 이상의 시간이 필요하다. 하나의 family가 아니라 전 단백질에 걸쳐 domain 분석을 하는 경우, 약 50000개의 단백질만을 대상으로 하는 경우에도 약 1000000배의 시간이 필요하다.

[0085] 또한, memory도 단백질의 수에 비례하여(실제로는 그 이상) 필요하게 된다. 따라서, 50개에 대해 초 단위의 실행시간이(현재 SGI의 서버급인 Origin 2000속도를 기준) 필요하다면, 50000개에 대해서는 몇 십일 혹은 몇 백일이 필요한 것이다.

[0086] 따라서, 현재의 알고리즘을 개선이 이루어져야 하고, 또한 대용량의 데이터 처리라는 면에서 개량할 필요성이 제기되는 것이다. 하나의 유전자 분석을 위해 데이터베이스를 대상으로 수십만의 서열(sequence) 정렬(Alignment)를 수행하는 것은 일반적으로 사용되고 있다. 한편, 상동성 분석을 통해 발견할 수 있는 기능 연관성은 일부에 지나지 않는다.

[0087] Weak 상동관계를 검출할 수 있는 열쇠는 scoring matrix에 있고, 이러한 scoring matrix의 개량과 서열(sequence) 정렬(Alignment) 알고리즘은 순환적으로 연관되어 있다. 유전체 연구의 병목이자 주 방법론이 될 유전자 분석의 효율성은, 가장 고전적인 정렬(Alignment) 알고리즘 이론의 개량과 응용 프로그램 수준에서의 알고리즘 개량이라는 두 가지 문제와 깊이 관련되어 있는 것이다

발명의 내용

해결하려는 과제

[0088] 본 발명이 해결하고자 하는 과제는 k-mer 거리와 병렬화를 통해 가장 많이 사용되고 있는 needlman-wunsch 알고리즘의 성능을 획기적으로 개선할 수 있는 FGA 알고리즘을 이용하는 서열 유사도 측정 시스템 및 이를 이용한 서열 유사도 측정 방법을 제안하고자 하는 것이다.

과제의 해결 수단

[0089] 상기 과제를 해결하기 위한 본 발명의 실시 예에 따른 FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템은 복수 개의 서열(sequence) 집합 데이터들이 저장된 데이터 베이스; 상기 데이터 베이스로부터 상기 복수 개의 서열(sequence) 집합 데이터들 중 2개의 서열(sequence) 데이터들 및 외부로부터 기설정된 기준값 데이터를 입력받아 상기 2개의 서열(sequence) 데이터들의 유사도 값이 상기 기준값 이상인 데이터 값을 추출하는 복수 개의 프로세서들; 상기 복수 개의 프로세서로들 각각으로부터 출력된 결과값들을 정렬한 후, 병렬화 처리하는 처리부를 포함하며, 상기 복수 개의 프로세서들 각각은, 상기 2개의 서열(sequence) 데이터를 간의 k-mer 거리를 계산하는 k-mer 알고리즘이 프로그래밍되어, 상기 2개의 서열(sequence) 데이터들의 k-mer 거리값인 제1 결과값을 출력하는 처리 모듈; 및

[0090] 상기 제1 결과값 및 상기 기준값 데이터를 수신하여 상기 제1 결과값 중 상기 기준값 이상의 데이터인 제2 결과값을 출력하는 판단 모듈;을 포함한다.

[0091] 상기 과제를 해결하기 위한 본 발명의 실시 예에 따른 FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템을 이용한 서열(sequence) 유사도 측정 방법은 데이터 베이스로부터 복수 개의 서열(sequence) 집합 데이터들 중 2개의 서열(sequence) 데이터들 및 외부로부터 기설정된 기준값 데이터를 입력받은 후, 상기 2개의 서열(sequence) 데이터들의 서열(sequence) 쌍에 따라 프로세서에서 분류하는 제1 단계; 상기 제1 단계 이후, k-mer 알고리즘이 적용된 상기 프로세서의 처리 모듈에서 상기 2개의 서열(sequence) 데이터들의 서열(sequence) 쌍의

k-mer 거리를 계산한 제1 결과값을 추출하는 제2 단계; 상기 제2 단계 이후, 상기 제1 결과값 중 상기 기준 데이터의 이상값인 제2 결과값을 상기 프로세서의 판단 모듈에서 추출하는 제3 단계; 및 처리부에서 상기 제3 단계의 결과값을 정렬화한 후, 병렬화 처리하는 제4 단계를 포함한다.

[0092] 상기 k-mer 알고리즘은, 상기 2개의 서열(sequence) 데이터들의 k-mer 프로파일과 서열(sequence)의 길이 및 k-mer 크기에 따른 값을 아래에 기재된 식에 적용하여 k-mer 거리를 도출하는 것을 특징으로 한다.

[0093] [식]

[0094] $$d_{X,Y} = \frac{\sum_{\tau} \min(n_X(\tau), n_Y(\tau))}{\min(\ell_X, \ell_Y) - k + 1}$$

[0095] 여기서, X, Y는 서열(sequence) 데이터를 의미하고, $d_{X,Y}$는 서열 X 및 서열 Y 사이의 k-mer 거리(distance)를 의미하며, τ는 k-mer를 의미하고, nx(τ)는 서열 X에서의 해당 k-mer가 존재하는 횟수를 의미하며, n_Y(τ)는 서열 Y에서의 해당 k-mer가 존재하는 횟수를 의미하고, lx는 서열 X의 전체길이를 의미하며, l_Y는 서열 Y의 전체길이를 의미하고, k는 k-mer의 길이를 나타내는 상수값을 의미한다.

[0096] 상기 제3 단계는 상기 두 서열(sequence) 간의 k-mer 거리 및 글로벌 정렬 유사도에서 추출된 유전적 거리 간의 상관관계를 도출하는 단계를 포함하는 것을 특징으로 한다.

발명의 효과

[0097] 본 발명에 따르면 k-mer 거리와 병렬화를 통해 가장 많이 사용되고 있는 need leman-wunsch 알고리즘의 성능을 획기적으로 개선할 수 있다.

[0098] 첫째, k-mer 거리 값을 이용하여 처리해야 할 글로벌 정렬 대상을 줄일 수 있다는 이점과 둘째, 병렬화를 통해 시스템 자원을 100% 활용한다는 이점과 셋째, 8개의 CPU에서 기존 글로벌 정렬 프로그램에 비해 최대 40배 이상의 우수한 성능 향상을 얻을 수 있다는 이점이 있다.

도면의 간단한 설명

[0099] 도 1은 본 발명의 실시 예에 따른 FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템을 나타낸 블럭도이다.

도 2는 도 1에 도시된 시스템을 이용한 서열(sequence) 유사도 측정 방법을 설명하기 위한 플로우 챠트이다.

도 3은 k-mer 거리와 유전적 거리(needle distance)와의 상관 관계를 나타낸 그래프이다.

도 4a는 본 발명의 FGA 저장구조를 나타낸 예시도이다.

도 4b는 FGA 데이터 저장 예제를 나타낸 예시도이다.

도 5는 FGA와 유전적 거리간의 처리 시간을 비교한 그래프이다.

도 6은 프로세스 개수에 따른 성능향상의 관계를 나타낸 그래프이다.

발명을 실시하기 위한 구체적인 내용

[0100] 본 명세서 또는 출원에 개시되어 있는 본 발명의 개념에 따른 실시 예들에 대해서 특정한 구조적 내지 기능적 설명들은 단지 본 발명의 개념에 따른 실시 예를 설명하기 위한 목적으로 예시된 것으로, 본 발명의 개념에 따른 실시 예들은 다양한 형태로 실시될 수 있으며 본 명세서 또는 출원에 설명된 실시 예들에 한정되는 것으로

해석되어서는 아니된다.

[0101] 본 발명의 개념에 따른 실시 예는 다양한 변경을 가할 수 있고 여러 가지 형태를 가질 수 있으므로 특정 실시 예들을 도면에 예시하고 본 명세서 또는 출원에 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시 예를 특정한 개시 형태에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0102] 제1 및/또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 이탈되지 않은 채, 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.

[0103] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는, 중간에 다른 구성요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0104] 본 명세서에서 사용한 용어는 단지 특정한 실시 예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 설시된 특징, 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0105] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 가지는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0106] 이하, 첨부한 도면을 참조하여 본 발명의 바람직한 실시 예를 설명함으로써, 본 발명을 상세히 설명할 것이며, 같은 문자는 같은 의미를 가진다.

[0107] 먼저 서열(sequence) 비교는 단백질 서열(sequence)의 기능이나 구조 분석에서 가장 기본적인 연산이다.

[0108] 서열(sequence)의 유사도는 두 서열(sequence)의 구조와 기능, 진화에 대한 정보를 제공한다. NGS와 같은 시퀀싱 기술의 발전으로 처리해야 할 서열(sequence)의 양이 급격히 증가하면서 서열(sequence) 정렬 성능이 이슈가 되고 있다.

[0109] 가장 대표적인 서열(sequence) 비교 알고리즘은 다이나믹 프로그래밍 기반의 글로벌 정렬과 로컬 정렬이다. 글로벌 정렬은 두 서열(sequence)을 전체적으로 비교하여 최대의 유사도를 찾아내며, 로컬 정렬은 두 서열(sequence) 간의 특정 부분에서 유사도가 있는지를 확인하기 위해 주로 사용한다. 가장 대표적인 글로벌 정렬 및 로컬 정렬 알고리즘으로는 needleman-wunsch 알고리즘과 smith-waterman 알고리즘이 있다.

[0110] 도 1은 본 발명의 실시 예에 따른 FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템을 나타낸 블럭도이다.

[0111] 도 1에 도시된 바와 같이, 본 발명의 시스템(100)은 데이터 베이스(110), 복수 개의 프로세서들(120) 및 처리부(130)를 포함한다.

[0112] 상기 데이터 베이스(110)는 복수 개의 서열(sequence) 집합 데이터들이 저장된다.

[0113] 상기 복수 개의 프로세서들 각각(120)은 상기 데이터 베이스로부터 상기 복수 개의 서열(sequence) 집합 데이터들 중 2개의 서열(sequence) 데이터들 및 외부로부터 기설정된 기준값 데이터를 입력받아 상기 2개의 서열(sequence) 데이터들의 유사도 값이 상기 기준값 이상인 데이터 값을 추출하는 기능을 수행한다.

[0114] 보다 구체적으로, 상기 복수 개의 프로세서들 각각(120)은 처리 모듈(121) 및 판단 모듈(122)을 포함한다.

[0115] 상기 처리 모듈(121)은 상기 2개의 서열(sequence) 데이터를 간의 k-mer 거리를 계산하는 k-mer 알고리즘이 프로그래밍되어, 상기 2개의 서열(sequence) 데이터들의 k-mer 거리값인 제1 결과값을 출력하는 기능을 수행한다.

[0116] 상기 판단 모듈(122)은 상기 제1 결과값 및 상기 기준값 데이터를 수신하여 상기 제1 결과값 중 상기 기준값 이상의 데이터인 제2 결과값을 출력하는 기능을 수행한

[0117] 상기 처리부(130)는 상기 복수 개의 프로세서(120)들 각각으로부터 출력된 결과값들을 정렬한 후, 병렬화 처리하는 기능을 수행한다.

[0118] 도 2는 도 1에 도시된 FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템을 이용한 서열(sequence) 유사도 측정 방법을 나타낸 블럭도이다.

[0119] 도 2에 도시된 바와 같이, 본 발명의 서열(sequence) 유사도 측정 방법(S100)은 제1 단계(S110) 내지 제4 단계(S140)를 포함한다.

[0120] 상기 제1 단계(S100)는 데이터 베이스로(110)부터 복수 개의 서열(sequence) 집합 데이터들 중 2개의 서열(sequence) 데이터들 및 외부로부터 기설정된 기준값 데이터를 입력받은 후, 상기 2개의 서열(sequence) 데이터들의 서열(sequence) 쌍에 따라 프로세서(120)에서 분류하는 단계일 수 있다.

[0121] 상기 제2 단계(S120)는, 상기 제1 단계(S110) 이후, k-mer 알고리즘이 적용된 상기 프로세서(120)에서 상기 2개의 서열(sequence) 데이터들의 서열(sequence) 쌍의 k-mer 거리를 계산한 제1 결과값을 추출하는 단계일 수 있다.

[0122] 상기 제3 단계(S130)는 상기 제2 단계(S120) 이후, 상기 제1 결과값 중 상기 기준 데이터의 이상인 제2 결과값을 상기 프로세서(120)에서 추출하는 단계일 수 있다.

[0123] 또한, 상기 제3 단계(S130)는 상기 두 서열(sequence) 간의 k-mer 거리 및 글로벌 정렬 유사도에서 추출된 유전적 거리 간의 상관관계를 도출하는 단계를 포함할 수 있다.

[0124] 상기 제4 단계(S140)는 처리부(130)에서 상기 제3 단계(S130)의 결과값을 정렬화하여, 병렬화 처리하는 단계일 수 있다.

[0125] 상기 k-mer 알고리즘은 상기 2개의 서열(sequence) 데이터들의 k-mer 프로파일과 서열(sequence)의 길이 및 k-mer 크기에 따른 값을 아래에 기재된 식에 적용하여 k-mer 거리를 도출한다.

[0126] [식]

$$d_{X,Y} = \frac{\sum_{\tau} \min(n_X(\tau), n_Y(\tau))}{\min(\ell_X, \ell_Y) - k + 1}$$

[0127]

[0128] 여기서, X, Y는 서열(sequence) 데이터를 의미하고, $d_{X,Y}$는 서열 X 및 서열 Y 사이의 k-mer 거리(distance)를 의미하며, τ는 k-mer를 의미하고, nx(τ)는 서열 X에서의 해당 k-mer가 존재하는 횟수를 의미하며, $n_Y(\tau)$는 서열 Y에서의 해당 k-mer가 존재하는 횟수를 의미하고, lx는 서열 X의 전체길이를 의미하며, l_Y는 서열 Y의 전체길이를 의미하고, k는 k-mer의 길이를 나타내는 상수값을 의미한다.

[0129] 이하에서는 본원발명에서 사용된 FGA의 성능 향상 핵심 원리인 k-mer 거리와 유전적 거리 간의 상관관계와 병렬화에 대해서 설명하도록 한다.

[0130] 먼저, FGA에서는 특정 cutoff의 유사도를 가지는 서열(sequence) 쌍을 찾기 위해 k-mer 거리를 이용한다.

[0131] k-mer 거리는 서열(sequence)정렬[], 계통분류분석[], 수평적 유전자 이동[]을 포함한 생명정보의 다양한 분야에서 유용하게 이용되는 검증된 기법이다. k-mer는 k-tuple 이라고도 하는데 k 개의 연속된 서열(sequence)로 구성된다. 예를 들어, ATGCATGC라는 서열(sequence)이 있을 때 k가 3이면 k-mer는 {ATG, TGC, GCA, CAT, ATG, TGC}가 된다 여기서 [{ATG:2}, {TGC:2}, {GCA:1}, {CAT:1}, {ATG:1}]과 같이 k-mer 별 빈도 수가 계산되는데 이를 k-mer 프로파일이라고 한다.

[0132] k-mer 거리는 두 서열(sequence)에서의 kmer 프로파일과 서열(sequence)의 길이 그리고 k-mer 크기를 이용하여 아래의 식을 통해 계산된다.

[0133] [식]

$$d_{X,Y} = \frac{\sum_{\tau} \min(n_X(\tau), n_Y(\tau))}{\min(\ell_X, \ell_Y) - k + 1}$$

[0134]

[0135] 여기서, X, Y는 서열(sequence) 데이터를 의미하고, $d_{X,Y}$는 서열 X 및 서열 Y 사이의 k-mer 거리(distance)를 의미하며, τ는 k-mer를 의미하고, nx(τ)는 서열 X에서의 해당 k-mer가 존재하는 횟수를 의미하며, $n_Y(\tau)$는 서열 Y에서의 해당 k-mer가 존재하는 횟수를 의미하고, lx는 서열 X의 전체길이를 의미하며, l_Y는 서열 Y의 전체길이를 의미하고, k는 k-mer의 길이를 나타내는 상수값을 의미한다.

[0136] 아래의 표 1은 k를 5라고 했을 때, 두 서열(sequence)에 대한 k-mer 거리를 계산하는 예제를 보여준다.

[0137] [표 1]

No	seq	len	knode
1	ATAATAATG	9	[*AATAA:1*], [*ATAAT:2*], [*TAATA:1*], {TAATG:1}]
2	AATAATACTAA	11	[*AATAA:1*], {AATAC:1},{ACTAA:1}, [*ATAAT:1*], {ATACT:1}, [*TAATA:1*], {TACTA:1}]

$min(l_{seq1}, l_{seq2})=9$

$\sum_{\tau} \min(n_{seq1(\tau)}, n_{seq2}(\tau))$= 1(AATAA) +1(ATAAT)+1(TAATA) = 3

$$\therefore d_{seq1,\,seq2} = \frac{3}{9-5+1} = 0.6$$

[0138]

[0139] 위에서 볼 수 있듯이 k-mer 거리를 위해서는 메모리 상에 프로파일을 관리하기 위한 별도의 공간(knode 필드)dl 필요하지만, 처리 시간은 k-mer 프로파일에서 k-mer 개수 민큼의 비교 연산으로써 글로벌 정렬의 에 비해 처리 속도가 상당히 빠름을 알 수 있다.

[0140] 도 3은 두 서열(sequence) 간의 k-mer 거리와 글로벌 정렬의 유사도에서 추출된 유전적 거리 간의 관계를 나타낸 그래프이다.(k=7)

[0141] 도 3을 참조하면, 두 서열(sequence)의 k-mer 거리와 유전적 거리는 연관성이 높음을 알 수 있다. 특히 사용자가 주로 관심이 있는 90% 이상의 유사도를 가지는 영역에서는 그 관계가 더욱 명확하다. 또한 90% 이상의 유사도를 가지는 서열(sequence) 쌍은 전체에서 1~8% 정도임을 확인할 수 있다.

[0142] 즉, k값 7에서 k-mer 거리 0.5를 기준으로 전체 연산의 90% 이상을 제거할 수 있음을 의미한다. FGA에서는 이러

한 대부분의 불필요한 연산을 제거함으로써 성능을 개선할 수 있다.

[0143] 도 4a는 FGA 저장구조를 나타내며, 도 4b는 FGA 데이터 저장 예제를 나타낸 예시도이다.

[0144] 이하에서는 도 4a 및 도 4b를 참조하여, 본 발명의 실시 예에 따른 FGA 알고리즘 처리 절차에 대해 설명하도록 한다.

[0145] 서열(sequence) 메모리 로드시에는 k-mer 거리를 구하기 쉬운 자료구조를 위해 도 4a와 같이 사용자 데이터 타입을 정의하게 된다. 이때, 서열(sequence)(서열(sequence)) 타입은 서열(sequence)정보를 저장하는 데이터 타입으로 서열(sequence)에 대한 기본 정보, k-mer 집합, 그리고 k-mer 프로파일 정보를 관리한다. 이때 k-mer 프로파일 정보는 KMER_PRO라는 타입을 정의하여 k-mer와 그 개수를 별도로 관리한다. 대용량 데이터에 대한 메모리 최적화를 위해 최대한 동적 메모리 할당 기법을 적용하였고, 서열(sequence) 구조체의 **kmer 필드는 knode 필드를 생성한 뒤 해제하여 메모리 사용량을 줄인다.(도 4a 및 도 4b 참조)

[0146] 이후 처리부(130)에서 글로벌 정렬 대상 리스트가 정해지면 N 개의 프로세스로 병렬화 처리를 수행한다.

[0147] 병렬화 수행시 두 서열(sequence) 집합이 동일한 경우에는 두 서열(sequence)이 동일한 경우(A-A)와 순서만 뒤바뀐 경우(A-B, B-A) 중 하나만 선택하여 처리함으로써 연산의 수를 줄인다. 이렇게 줄인 경우 연산의 횟수가 $n(n-1)/2$로써 n^2에 비해 반 이상 줄어들게 되며, 이 대상에 대해 글로벌 정렬 이후 사용자가 제시한 Cutoff를 만족하는 서열(sequence) 쌍에 대해서만 결과를 취합하여 제공하게 된다.

[0148] 도 5는 FGA와 유전적 거리간의 처리 시간을 비교한 그래프이며, 도 6은 프로세스 개수에 따른 성능향상의 관계를 나타낸 그래프이다.

[0149] 이하에서는 FGA의 성능 평가를 위해 현재 가장 많이 사용하고 있는 emboss 패키지 내의 needleall 프로그램을 활용하였다.

[0150] [표 2]

구분	설명
Program	*needleall* in Emboss Package
CPU	Intel(R) Xeon(R) CPU X5355 @ 2.66GHz * 8 cores
Memory	16,440,060 kB
OS	CentOS release 5.4 (Kernel 2.6.18-164.11.1)
Dataset	1000 sequences (400bp~500bp)

[0151]

[0152] [표 2]는 성능평가에 활용된 시스템 환경을 보여준다. 본 실험은 리눅스 배포판인 CentOS 상에서 8개의 CPU가 장착된 서버 환경에서 실험을 수행하였다. 이때 데이터 셋은 1000개의 서열(sequence)로써 1,000,000(1,000*1,000) 번의 글로벌 정렬이 필요하다.

[0153] 또한, FGA와 needleall 간의 성능 비교를 위해 두 가지의 평가 기준인 처리 시간과 성능 향상을 사용하였다. 처리 시간은 각 프로그램 별로 1000개의 서열(sequence)를 포함하는 두 그룹을 프로세서 개수에 따른 처리 속도를 측정하였다. 1,000,000 정렬을 8 개의 프로세서로 병렬처리 하면서 FGA와 needleall 간의 처리시간을 비교하였다.

[0154] 또한, needleall 처리 시간 대비 FGA 성능향상 정도를 계산하였는데 두 경우로 나누었다. 동일하게 병렬 처리를 한 경우(speeup1)와 그렇지 않은 경우(speedup2)로 나누어 성능향상정도를 확인하였다.

[0155] 아래의 표 3은 글로벌 정렬 처리 시간 비교를 나타낸 표이며, 표 4는 글로벌 정렬 성능향상 비교를 나타낸 표이다.

[0156] [표 3](단위; 초)

	1	2	3	4	5	6	7	8
FGA	92	53	34	28	22	19	16	16
needleall	3,636	1,956	1,308	977	820	700	614	540

[0157]

[0158] [표 4]

	1	2	3	4	5	6	7	8
Speedup1	39.52	36.91	38.47	34.89	37.27	36.84	38.38	33.75
Speedup2	39.52	68.60	106.94	129.86	165.27	191.37	227.25	227.25

[0159]

[0160] 본 발명에서는 대용량 서열(sequence) 집합 간의 글로벌 정렬 성능을 개선한 FGA 시스템을 제안하였으며, 성능을 개선하기 위해 k-mer 거리를 활용하여 처리 범위를 90% 이상 줄였고, 병렬화를 통해 시스템 내의 프로세서를 100%를 활용할 수 있도록 하여 기존 글로벌 정렬 프로그램에 비하여 40 배 이상 뛰어난 실험 결과를 얻을 수 있었다. 향후에는 FGA 시스템을 분산환경으로 확장하여 대용량 글로벌 정렬 서비스로 제공할 수 있을 것이다.

[0161] 또한, 본 발명에 따르면, k-mer 거리와 병렬화를 통해 가장 많이 사용되고 있는 need leman-wunsch 알고리즘의 성능을 획기적으로 개선할 수 있다.

[0162] 첫째, k-mer 거리 값을 이용하여 처리해야 할 글로벌 정렬 대상을 줄일 수 있다는 이점과 둘째, 병렬화를 통해 시스템 자원을 100% 활용한다는 이점과 셋째, 8개의 CPU에서 기존 글로벌 정렬 프로그램에 비해 최대 40배 이상의 우수한 성능 향상을 얻을 수 있다는 이점이 있다.

[0163] 이상과 같이 본 발명에 따른 FGA 알고리즘이 적용된 서열(sequence) 유사도 측정 시스템 및 이를 이용한 서열(sequence) 유사도 측정 방법을 도시한 도면을 참조하여 설명하였으나, 본 명세서에 개시된 실시예와 도면에 의해 본 발명이 한정되는 것은 아니며, 본 발명의 기술사상 범위에서 당업자에 의해 다양한 변형이 이루어질 수 있음은 물론이다.

부호의 설명

[0164] 100 : 시스템

110: 데이터 베이스

120: 프로세서

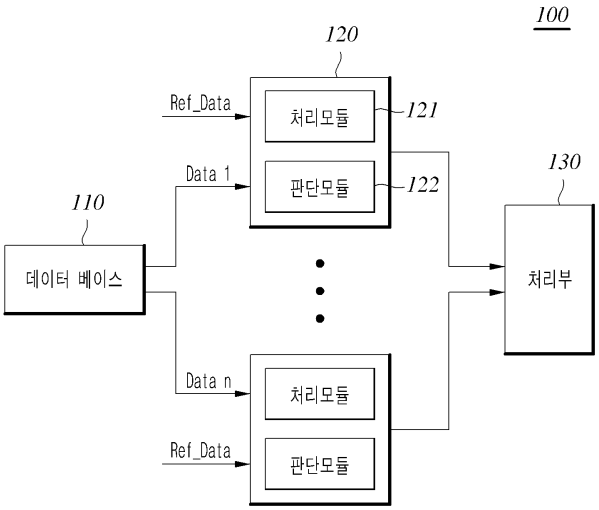
121: 처리 모듈

122: 판단 모듈

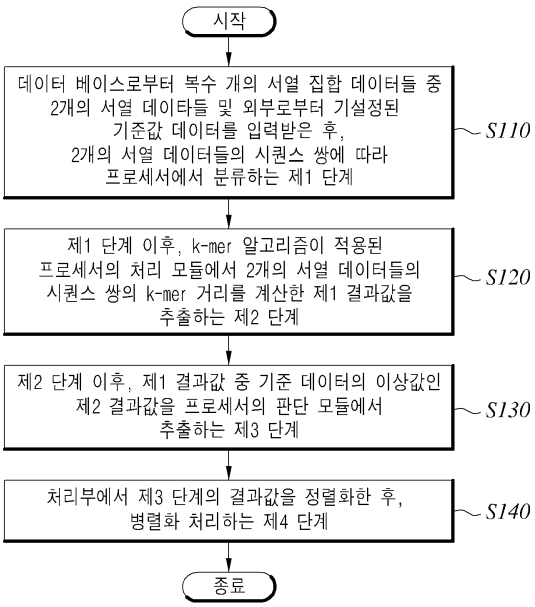
130: 처리부

도면

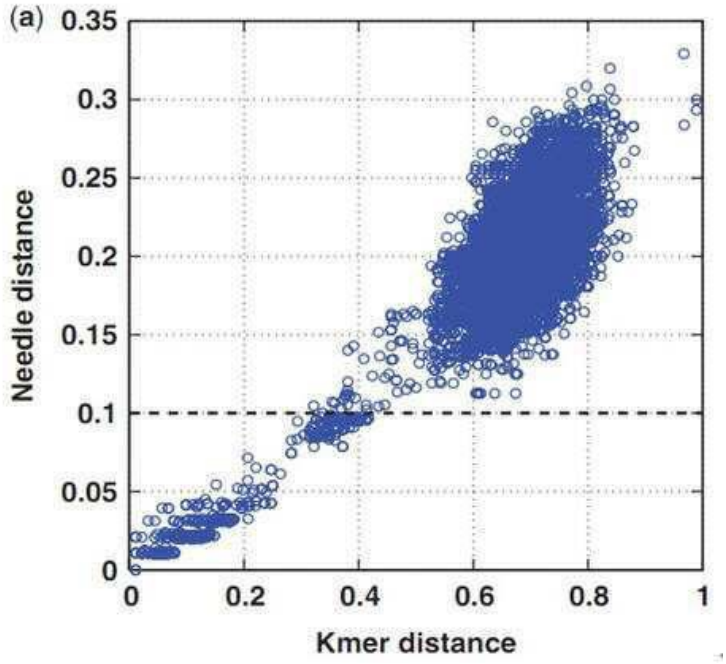
도면1


100
120
Ref_Data
처리모듈
121
Data 1
판단모듈
122
130
110
데이터 베이스
처리부
Data n
처리모듈
Ref_Data
판단모듈

도면2


시작
데이터 베이스로부터 복수 개의 서열 집합 데이터들 중 2개의 서열 데이타들 및 외부로부터 기설정된 기준값 데이터를 입력받은 후, 2개의 서열 데이터들의 시퀀스 쌍에 따라 프로세서에서 분류하는 제1 단계
S110
제1 단계 이후, k-mer 알고리즘이 적용된 프로세서의 처리 모듈에서 2개의 서열 데이터들의 시퀀스 쌍의 k-mer 거리를 계산한 제1 결과값을 추출하는 제2 단계
S120
제2 단계 이후, 제1 결과값 중 기준 데이터의 이상값인 제2 결과값을 프로세서의 판단 모듈에서 추출하는 제3 단계
S130
처리부에서 제3 단계의 결과값을 정렬화한 후, 병렬화 처리하는 제4 단계
S140
종료

도면3



도면4a

```
/* Kmer Profile Structure */
struct kmer_profile{
  double count;   // the number of kmer
  char *kmer;     // kmer
};
typedef struct kmer_profile KMER_PRO;

/* Sequence Data Structure */
struct sequence{
  int no;
  char *header;
  char *seq;
  int seq_len;
  char **kmer;
  int kmer_cnt;
  KMER_PRO *knode;
  int knode_count;
};
typedef struct node SEQUENCE;

SEQUENCE *seq1, *seq2;
```

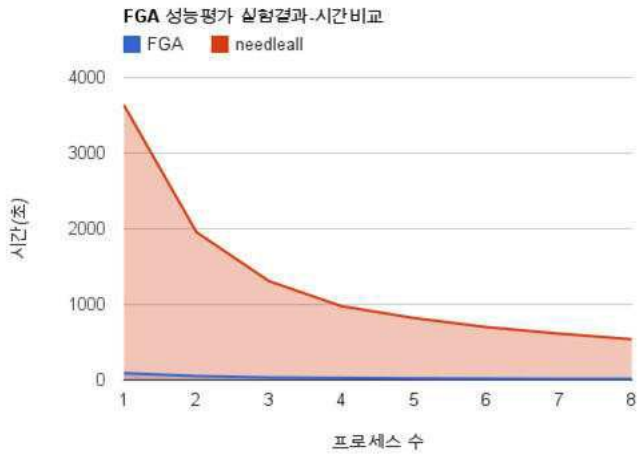
도면4b

No	1
header	Header1
Seq	ATAATAATG
Seq_Len	9

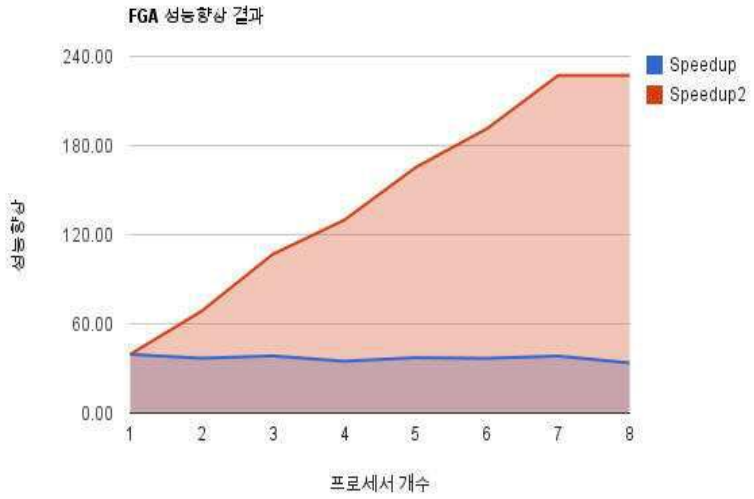
Kmer	{
	ATAAT,
	TAATA,
	AATAA,
	ATAAT,
	TAATG
	}

Kmer count	5
Knode	[{AATAA:1}, {ATAAT:2}, {TAATA:1}, {TAATG:1}]
Knode count	4

도면5



도면6


FGA 성능향상 결과
240.00
180.00
120.00
60.00
0.00
성능향상
1
2
3
4
5
6
7
8
프로세서 개수
Speedup
Speedup2