Informe del
Curso Optativo
de Algoritmos de Similitud
de Textos

Tema: Algoritmo de Smith-Waterman

Integrantes: Davier Sánchez Bello
Manuel Alejandro Gamboa Hernández

Introducción

La alineación de secuencias biológicas, como secuencias de ADN o proteínas, es un proceso crucial en bioinformática y biología computacional. Sirve para comparar secuencias relacionadas e inferir varios hechos sobre ellas, como una ascendencia evolutiva común o un funciones estructurales comunes. Las secuencias de ADN y proteínas evolucionan a través de un proceso de mutación y selección natural. Al comparar dos secuencias, podemos determinar si tienen un origen evolutivo común en caso de que su similitud es improbable que se deba al azar.

Existen dos tipos principales de alineación de secuencias: global y local.

- La alineación local, por otro lado, es más útil para secuencias disímiles que se sospecha contienen regiones de similitud o motivos de secuencia similares dentro de su contexto de secuencia más amplio. El algoritmo de Smith-Waterman es un método de alineación local general basado en el mismo esquema de programación dinámica pero con opciones adicionales para comenzar y terminar en cualquier lugar. Sobre este algoritmo estaremos hablando en este informe.

- La alineación local, por otro lado, es más útil para secuencias disímiles que se sospecha contienen regiones de similitud o motivos de secuencia similares dentro de su contexto de secuencia más amplio. El algoritmo de Smith-Waterman es un método de alineación local general basado en el mismo esquema de programación dinámica pero con opciones adicionales para comenzar y terminar en cualquier lugar. Sobre este algoritmo estaremos hablando en este informe.

Algoritmo Smith-Waterman

El algoritmo de Smith-Waterman se deriva del algoritmo de Needleman-Wunsch pero en lugar de trabajar con la secuencia entera, este algoritmo compara segmentos de todos los tamaños posibles, optimizando la medida de similaridad. Alinea dos cadenas usando sustituciones (matches/missmatches) o la colocación de espacios (insertions/deletions) representados por guiones.

Fue presentado por primera vez por T.F.Smith y M.S.Waterman en 1981 en “La Jornada de Biología Molecular”, en

una carta a la editorial, que consta de 3 páginas publicadas íntegramente. En el documento plantean que el objetivo que persiguen es la discusión de un algoritmo para encontrar un par de segmentos, uno por cada secuencia, tales que no haya otro par de segmentos con una similitud mayor.

Definen la formula recursiva usada en la programación dinámica de la siguiente forma:

Sean dos secuencias $A = a_1, a_2, \ldots, a_n$ y $B = b_1, b_2, \ldots, b_m$. Sea $s(a, b)$ la función de similitud entre dos elementos de las secuencias. Sea el vector W, tal que W_i tiene el peso de colocar un espacio de longitud i. Sea la matriz H, tal que H_{ij} contiene la máxima similitud de dos segmentos cualesquiera que terminan en a_i o b_j, donde $H_{k0} = H_{0l} = 0$, $0 \leq k \leq n$, $0 \leq l \leq m$. Estos valores son obtenidos por la relación:

$$H_{ij} = max\{ H_{i-1, j-1} + s(a_i, b_j), max_{(k \geq 1)} \{ H_{i-k, j} - W_k \}, max_{(l \geq 1)} \{ H_{i, j-l} - W_l \}, 0 \}, \quad (1)$$

$$1 \leq i \leq n, 1 \leq j \leq m.$$

De esta manera se cubre cada posible segmento que termina en a_i y en b_j, incluidos aquellos que terminan con una inserción de espacios vacíos de tamaño k. Añaden que para encontrar el par de segmentos con la mayor similitud, primero que debe tomarse el máximo elemento de H y luego realizar un rastreo desde la posición de mayor similitud hasta una posición donde $H_{ij} = 0$. Entonces los segmentos serian los que van entre ambas i y ambas j respectivamente. Para encontrar el segundo segmento con mayor similitud bastaría buscar el segundo mayor valor de H y realizar el mismo procedimiento de rastreo pero evitando pasar por donde ha pasado el rastreo para encontrar el mayor elemento.

El rastreo procede de la siguiente manera: si se encuentra en una casilla retrocede a la casilla que aportó el valor a dicha casilla.

Respecto a la manera en que se asignan costos a los espacios, tenemos que en la biología molecular es mucho más común las eliminaciones de segmentos prolongados en las cadenas que la aparición esporádica de eliminaciones puntuales. Por tanto los sistemas que se usan para puntuar la aparición de espacios en el algoritmo Smith-Waterman generalmente sigue uno de los siguientes modelos:

- Modelo de huecos afines: otorga un valor fijo a la creación de un hueco, y luego, una penalización más pequeña por la ampliación de este.
- Modelo de huecos convexos: a diferencia del modelo de huecos afines donde los costos de inicializar y extender un hueco son constantes, en la penalización de huecos convexos el costo de extender el hueco disminuye a medida que aumenta su longitud, o sea :

Complejidad Temporal

En cuanto a la complejidad temporal del algoritmo, esta es $O(nm^2)$, donde el cuello de botella se encuentra en rellenar la matriz *H*. Es más complejo que el algoritmo de Needleman-Wunsch que tiene complejidad $O(mn)$. Esta complejidad se vuelve un obstáculo para cadenas demasiado grandes, pero las cadenas biológicas se caracterizan precisamente por su gran tamaño. Por tal motivo, existen numerosas variaciones del algoritmo que buscan acelerar la constante con la que se realiza el cómputo.

Algunas de estas variaciones son modificaciones para adaptarlo específicamente a hardware que trabaja con paralelismo o supercomputadoras de velocidad muy elevada que permiten incrementar la eficiencia del algoritmo entre x15 y x100 veces. Entre ellas podemos mencionar el uso de chips de campos programables FPGA, o el uso de unidades de procesamiento grafico (GPU) para correr el algoritmo.

Otras modificaciones del algoritmo lo han llevado a O(mn) como es el caso de las modificación presentada por el japonés Osamu Gotoh precisamente en la revista “Journal of Molecular Biology” en 1982, o el refinamiento de este último, en cuanto a correctitud, presentado por Stephen Altschul en el Boletín de Biología Matemática en 1982.

La complejidad en cuanto a uso de memoria del algoritmo es $O(mn)$, pero fue optimizada, para el caso en el cual solo se desea encontrar un alineamiento óptimo, hasta $O(n)$ donde *n* es la menor de las cadenas, por Eugene Myers y Webb Miller en 1988. Esta optimización también esta asentada sobre la realizada previamente por Gotoh.

Existen implementaciones del algoritmo de Smith – Waterman en toda clase de lenguajes. Por ejemplo en *python* hay un modulo llamado *swalign* que realiza el cálculo de la similitud de dos strings acorde a Smith-Waterman y realiza la alineación correspondiente. En R en la biblioteca BioStrings, cuando al método *pairwiseAligment* se le pasa el parámetro “local”, esta función realiza la alineación acorde al algoritmo Smith – Waterman.

Descripción de la implementación en c++

Aunque aún no se ha llevado a una biblioteca la implementación es cuestión de muy poco tiempo que esto suceda. La clase *SWCalculator* es donde se realiza el alineado de las cadenas. Tiene dos métodos esenciales, el *fill_matrix* y el *traceback*; aunque ambos son privados, siendo públicos únicamente el método *Alinear*, y un método para imprimir la matriz generada por el algoritmo Smith_Waterman.

La clase *SWCalculator* es extensible, desde el momento en que toma en su constructor un delegado de función para que el usuario pase el método para calcular el peso de los huecos en la cadena de la forma que le parezca correcta. Además en su constructor el objeto *SWCalculator* recibe una matriz formulada como un mapa de c++, con los pesos asignados a cada uno de los posibles match o missmatch entre los caracteres de las cadenas. Esta clase está pensada para que una misma instancia pueda realizar cuantos cálculos se necesiten para cadenas diferentes, pero, siempre manteniendo los mismos parámetros de match/missmatch así como de hueco de cadena. O sea, a cada instancia de *SWCalculator* se le asocian los parámetros con los que realizara sus cálculos y a partir de entonces sirve como un motor de cálculo de alineamiento para pares de cadenas usando los parámetros dados en su construcción. Si se desea cambiar de parámetros, sencillamente se crea otro objeto SWCalculator y listo.

El *enum direction* sirve para poder recordar para cada casilla de la matriz de donde vino su valor. Mediante un trabajo de álgebra booleana con el operador '|', mantenemos un arreglo paralelo al de la matriz de los scores, y es el de la matriz de direcciones. Las direcciones son números entre 0 y 7. Si una casilla tiene una dirección con el bit menos significativo encendido esto significa que tomo su valor de la casilla inmediata superior, si tiene el bit del medio encendido significa que tomó su valor de la diagonal y si tiene su bit más significativo encendido significa que lo tomó de la casilla a su izquierda. Por supuesto, en los casos de empate, o sea, en los casos en que una casilla bien puede haber tomado su valor de dos o tres de las direcciones, debemos preservar todos estos posibles rumbos para dejar que el usuario sea el que seleccione cuál es el más representativo biológicamente hablando. Es por ello que se definió la operación de 'or' booleano '|' entre direcciones, que funciona idénticamente a la operación de 'or' booleano entre enteros. Esta operación permite preservar todas las direcciones de donde puede haber venido el valor de la casilla.

El método *fill_matrix* rellena la matriz siguiendo al pie de la letra la descripción del algoritmo. No solo rellena una matriz de score (llamada *H* en la implementación) sino que rellena una matriz de direcciones D. El método *trace_back* realiza una búsqueda en profundidad partiendo de las casillas de la matriz donde se obtuvo la mayor puntuación, moviéndose según las direcciones autorizadas en

D, que comprueba haciendo uso del operador booleano & (comprobando que bits de la dirección están encendidos) y deteniéndose al encontrar una casilla de score 0.

El método *trace_back* además rodea los segmentos de cadena emparejados con las restantes partes de cada cadena, de manera que podemos ver como queda la alineación de manera global entre las dos cadenas. La parte alineada de ambas cadenas se marca en verde.