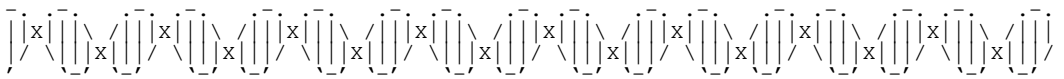
# Comparación de Secuencias Biológicas
## Algoritmia

Oswaldo Trelles
< ots@ac.uma.es >

Departamento de Arquitectura de Computadores, U.Málaga
Asociado a la Unidad de Biocomputación (CNB)



Se ofrece una revisión de la comparación de seuencias biológicas centrada en los aspectos formales, algorítmicos y computacionales. Los problemas *computacionales* que emergen del área de análisis de secuencias, la hacen interesante para un área de Arquitectura de Computadores cuyos desarrollos abordan la búsqueda de soluciones para problemas intensivos en demanda de recursos (CPU, E/S, memoria, etc). En cualquier caso los conceptos con significado biológico que se mencionan deberán ser contrastados con la contraparte biológica para darles unidad. En nuestro caso, esta consolidación se cimienta en la colaboración con grupos de biología (entre ellos, BioComputación y Diseño de Proteinas del CNB), con los cuales formamos una unidad asociada.

## *Indice*

- Aspectos Generales
- Comparación por identidades
- Complejidad Algorítmica
- Comparación por semejanza.
- Interrupciones en el alineamiento
- Algoritmos (Prog.Dinámica, FASTA, BLAST)
- Alineamientos múltiples
- Comentarios finales

---

Se revisan de conceptos básicos sobre lo que es una secuencia de símbolos y el caso particular de símbolos biológicos y lo que representa comparar/alinear secuencias. Se utiliza una aproximación incremental que parte de un algoritmo simple para introducir conceptos de semejanza, y discurre añadiendo nuevas necesidades para mejorar la comparación mediante los conceptos de matrices de semejanza, interrupciones, alineamientos locales y globales, y también se revisan los aspectos de complejidad computacional.

Luego se aborda una revisión de los algoritmos actuales más usados para la comparación de secuencias biológicas y se extiende el caso de parejas de secuencias a varias secuencias para hablar de alineamientos múltiples y se concluye con algunos comentarios sobre lo expuesto.

# Conceptos Generales

**Secuencia** Cadena lineal, finita y ordenada de símbolos pertenecientes a un alfabeto. El número de símbolos de la cadena representa su longitud

**Alfabeto** Conjunto de símbolos diferentes usados para representar las secuencias.

*ADN* $\boldsymbol{A}$ = { a,c,g,t|u }

*Proteínas* $\boldsymbol{A}$ = { a,c,d,e,f,g,h,i,k,l,m,n,p,q,r,s,t,v,w,y }

ADN y proteinas son secuencias co-lineales.

Existen diversos alfabetos (carácter químico, funcional, estructural, etc.) en función de lo que se desea expresar y mecanismos de traducción entre alfabetos.

**Comparación** El objetivo (inicial y algorítmico) de comparar dos secuencias es encontrar la posición relativa de ambas en las que se produzca mayor número de coincidencias entre sus componentes.

El valor "número de coincidencias o identidades" representaría la valoración de su parecido y podría darse en valores absolutos o porcentuales (normalizado) dividiéndolo entre el máximo valor posible (la longitud de la secuencia más corta).

[*Nota*: el objetivo *biológico* de la comparación de secuencias se suele expresar en términos de inferencia de relaciones estructurales, funcionales o evolutivas entre las secuencias]

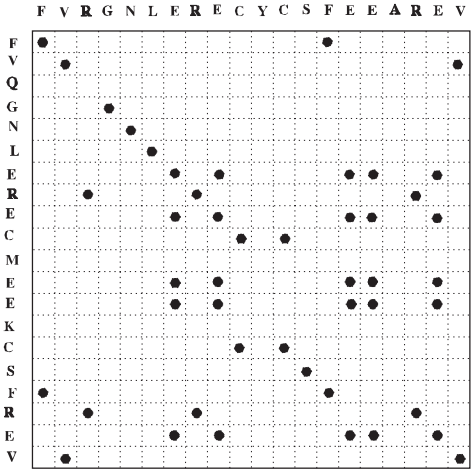
## Comparación básica por Identidades

```
                X:    TCAGACGATTG (n=11)
                Y:    ATCGGAGCTG  (m=10)

         TCAGACGATTG   (0)                          •
ATCGGAGCTG                                          •
                                                    •
         TCAGACGATTG   (1)                  TCAGACGATTG          (3)
 ATCGGAGCTG                                        ATCGGAGCTG

         TCAGACGATTG   (0)                  TCAGACGATTG          (1)
  ATCGGAGCTG                                         ATCGGAGCTG

         TCAGACGATTG   (2)                  TCAGACGATTG          (0)
   ATCGGAGCTG                                          ATCGGAGCTG

         TCAGACGATTG                        TCAGACGATTG          (0)
    ATCGGAGCTG         (0)                             ATCGGAGCTG
             •
                          -------------------------------
```

*Algoritmo*: Desplazar una de las secuencias debajo de la otra y contabilizar los residuos que coinciden. El mejor alineamiento corresponde a la posición en que éste valor es máximo (indicado al lado derecho entre paréntesis)

# Comparación básica por Identidades

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | T | C | A | G | A | C | G | A | T | T | G |
| -1 | A | | | 1 | | 1 | | | 1 | | | |
| -2 | T | 1 | | | | | | | | 2 | 1 | |
| -3 | C | | 2 | | | | 1 | | | | | |
| -4 | G | | | | 1 | | | 2 | | | | 3 |
| -5 | G | | | | 3 | | | 1 | | | | 1 |
| -6 | A | | | 1 | | 4 | | | 2 | | | |
| -7 | G | | | | 2 | | | 2 | | | | 2 |
| -8 | C | | 1 | | | | 1 | | | | | |
| -9 | T | 1 | | | | | | | | 3 | 1 | |
| -10 | G | | | | 2 | | | 3 | | | | 2 |




"*Desplazar las secuencias*" equivale a colocar una de ellas en vertical, la otra en horizontal y recorrer cada una de las diagonales de la matriz así formada, acumulando el número de coincidencias que se produzcan (figura izquierda). La diagonal en la que se produzca mayor número de identidades representará el desplazamiento relativo que mejor alinea las secuencias. El número de identidades que se contabilicen en dicha diagonal valorará el parecido (por identidades) entre las secuencias. Si representamos las coincidencias por un punto (en la figura de la derecha) obtendremos lo que se denomina un DotPlot o matriz de puntos (Maizel & Lenk, 1981), en el que los fragmentos diagonales que se forman ofrecen información *visual* acerca de las relaciones entre las secuencias en comparación.

Podemos aquí introducir *intuitivamente* el concepto de complejidad algorítmica que está relacionado al número de operaciones que requiere la solución de un problema. En este caso el número de operaciones será *proporcional* al tamaño de la matriz formada por las dos secuencias (NxM siendo N y M las longitudes de las secuencias). Se escribe $O(N^2)$ si se asume que N y M son en promedio iguales (cuando se realizan muchas comparaciones).

*Notas*

[1] Al recorrer las diagonales se ha ido acumulando en cada celda el número de coincidencias en la diagonal. En este caso, el mejor alineamiento se produce en la diagonal -1 y alcanza un puntaje de 4, que representaría un 40% de semejanza (sobre un máximo de 10 coincidencias).

[2] La numeración de las diagonales puede obtenerse restando la posición del primer residuo de cada secuencia (la diagonal -1, indicaría que se ha alineado el primer residuo de la secuencia horizontal con el segundo residuo de la secuencia vertical (1-2=-1))

[3] En este ejercicio, el mejor alineamiento por identidades y sin interrupciones es:

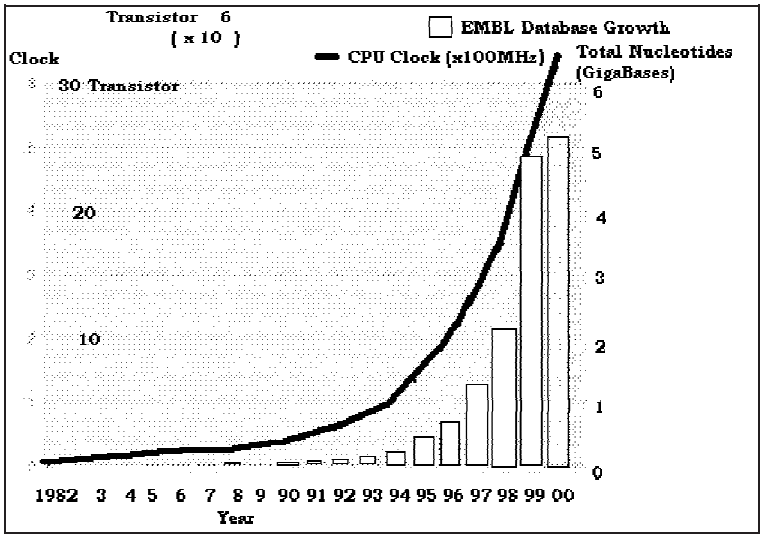
```
pos:       12345678901
     X:    TCAGACGATTG     (n=11)
           || ||
     Y:  ATCGGAGCTG        (m=10)
pos:     1234567890
```

# Volumen de datos
# ...necesidad de velocidad de proceso

El crecimiento de los datos referidos a las secuencias biológicas es espectacular. Así, la primera proteína secuenciada (la insulina humana, 50 aa., Sanger, 1953), requirió, con la tecnología disponible en los años 50 alrededor de 10 años, en los 60 este trabajo podría haber requerido 3 años, en los 70 sólo 1 año y en los 80 una semana. Hoy en día se secuencia entre 1000 y 5000 nucleótidos por día, esperándose alcanzar del orden de $10^6$ n/día en el futuro cercano.

Actualmente (18.Enero.2000) la *European Molecular Biology Laboratory (EMBL) Nucleotide Sequence Database* (http://www.ebi.ac.uk/embl/index.html) contiene más de 5.3 millones de entradas que representan más de 5 GigaBytes de nucleótidos (http://www3.ebi.ac.uk/Services/DBStats). El crecimiento ha sido especialmente espectacular en los últimos tres años en los que el secuenciamiento de genomas ha producido un incremento sustancial de datos (nuevos y completos gemomas). En los primeros 8 meses de 1999 se anotaron más de 1.6 millones de entradas nuevas (más de 1.3 GigaBytes) con una tasa media de 6.400 entradas (5.4 MegaBases) por día.

Crecimiento de la BD del EMBL y de los recursos computacionales.




Por otra parte, los recursos computacionales también crecen con tasas exponenciales (otra de las analogías que se suelen encontrar entre computación y biología). Así, la potencia de CPU de los microchips (para tomar en cuenta algo de dominio común) de los PC's ha saltado desde los 4,77 MHz de los primeros Intel 8088 hasta los más de 500 MHz que son frecuentes hoy en día en los Pentium (por ejemplo en el Pent III-Xeon, Celeron, Itanium y similares). Es más, a inicio de año se anunciaba la primera CPU que superaría la barrera de los GigaHz de frecuencia de reloj (mas detalles en http://www.prisma.uvsq.fr/mirror/CIC/summary/local)

Otros recursos computacionales que también sirven de referencia acerca de la tasa de crecimiento de la potencia de cálculo son: el número de transistores que contiene la CPU, que ha pasado de los 0,029 x $10^6$ a los más de 30 millones actuales, velocidad de acceso a memoria, *caches* intermedias, etc.

Este aumento de la potencia de cálculo nos permite abordar problemas cada vez más grandes y de mayor complejidad. Sin embargo,no es suficiente para cubrir la demanda que generan muchas de las aplicaciones de áreas como el análisis de secuencias biológicas. Por esta razón, entre otras, es necesario optimizar los algoritmos, en la intención de cerrar esta brecha tecnológica.

# Reducción de la Complejidad Algorítmica

Una vez identificado el objetivo y el método para resolver una tarea, se aborda una labor importante desde el punto de vista computacional, que es la reducción de los tiempos de cálculo (optimización). Una estrategia muy utilizada cuando se realizan comparaciones masivas es la de organizar los datos en tablas de dispersión para que sean fácil y rápidamente localizables.

Por ejemplo, si tenemos la secuencia que se ha utilizado en los casos anteriores,

```
posición   :  12345678901
secuencia X:  TCAGACGATTG    (n=11)
```

su tabla de dispersión sería la siguiente:

| | |
|---|---|
| A | 3, 5, 8 |
| C | 2, 6 |
| G | 4, 7, 11 |
| T | 1, 9, 10 |

que consiste en una lista de las posiciones en que aparece cada elemento del alfabeto de la secuencia. Si ahora necesitamos comparar (por identidades) esta secuencia X, contra la siguiente secuencia Y, bastaría con acumular en la diagonal correspondiente el número de coincidencias (el número de la diagonal se obtiene restando las posiciones relativas de los residuos).

```
posición   :     1234567890
secuencia Y:     ATCGGAGCTG      (m=10)
```

Así, el primer residuo de la secuencia Y (posición 1, residuo A), producirá una coincidencia en cada una de las siguientes diagonales: 2 (3-1), 4 (5-1) y 7 (8-1). El segundo residuo de la secuencia Y (posición 2, residuo T), producirá una coincidencia en las diagonales: -1 (2-1), 7 (9-2) y 8 (10-2), y así sucesivamente.

Ahora el algoritmo requiere recorrer únicamente una vez la secuencia Y, y por cada elemento de ella, tantas posiciones como se indique en la tabla de dispersión de la secuencia X para dicho elemento (si se considera, de forma gruesa, que todos los elementos del alfabeto están igualmente representados, la longitud de una entrada de la tabla de dispersión contendrá en promedio $n/l$ elementos (siendo $n$ la longitud de la secuencia y $l$ la longitud de la tabla). En este caso la longitud de la tabla de dispersión es igual a la longitud del alfabeto, puesto que se ha dispersado a nivel de residuo).

Si elaboramos tablas de dispersión por palabras de longitud $k$ (*k-tuplas*), el número de entradas de la tabla crece ($l^k$) y por tanto, el número de elementos en cada entrada disminuye. Para el caso de proteinas, en que $l=20$, usando $k=2$, tendríamos 400 ($20^2$) elementos. Si consideramos que la longitud media de una proteína es de 400 aa (360 en la swissprot 36), en promedio tendríamos un valor en cada entrada, y por tanto, la complejidad algorítmica se aproximaría a la *suma* de las longitudes de las secuencias $O(N+M)$, en vez del producto, lo cual permite ir mucho más rápido.

# Comparación por Semejanza

El uso de identidades, con ser el punto de partida, es muy limitado y se muestra insuficiente para establecer la relación entre secuencias (Doolittle, 1981), entre otras cosas porque:

- El código genético es degenerado o redundante ($4^3$=64 codones -> 20 aa), y todos los aa (excepto triptófano y metionina), son especificados por dos o más codones (*i.e,* leucina, arginina y serina son especificados cada uno por seis posibles codones). Por tanto, un cambio a nivel genético puede no alterar la proteína especificada.

- La sustitución de un aminoácido por otro de propiedades similares (tamaño, carga, propiedades químicas, etc.), puede no tener gran influencia sobre la función global de la proteína.

- Es necesario considerar la inserción y la pérdida (*delección*) de residuos.

- Es necesario considerar la *cantidad de información* que acarrea cada símbolo del alfabeto (asociada a la frecuencia de aparición del símbolo)

- Es necesario incorporar el conocimiento *biológico* en el método de valoración. Por ejemplo, considerar las coincidencias CyS-Cys mas importantes que, digamos, las de Ala-Ala. O en el caso de ADN, se debería tener en cuenta que el hecho de alinear una A con una G (ambas purinas) o una C con una T (ambas pirimidinas) debería tener mejor valoración que el hecho de alinear una purina con una pirimidina, aunque esta puntuación deba ser menor que el hecho de alinear dos residuos iguales.

Por ello, las ideas actuales para comparar secuencias biológicas se basan en determinar una cierta función *distancia* entre las dos secuencias, que pretende informar acerca de la *cercanía* entre dos secuencias (en dependencia de los criterios de evaluación que se utilicen, esta será una distancia evolutiva, estructural, etc.). Por ejemplo, la *distancia* (Sellers, 1974) puede representar el mínimo coste de transformar la secuencia X en la Y por medio de la aplicación de una serie de transformaciones (sustituciones, inserciones, delecciones), cada una de las cuales tiene asociada un coste.

Para cada pareja de residuos $(a_i,a_j) \in \mathcal{A}^2$ existe un esquema de pesos $\mathbf{w}(a_i,a_j)$ que establece la relación entre ambos elementos del alfabeto. Este esquema de pesos varía desde modelos simples que asignan el valor 1 si $a_i = a_j$ y 0 si $a_i \neq a_j$, hasta modelos más complejos evolutivos, estructurales y/o funcionales que fijan un determinado costo por sustituir $a_i$ por $a_j$ dependiendo explícitamente tanto de *i* como de *j* y de su entorno.

# Mecanismos de puntuación

Las tablas de puntuación pretenden capturar el significado biológico de la semejanza entre residuos considerando reemplazos conservativos y diferencias en las frecuencias observadas de los mismos. Tanto en forma teórica (Altschul, *et al.* 1991) como práctica se observan mejores resultados cuando la tabla refleja el patrón de equivalencia de residuos que se da en secuencias relacionadas estructural, funcional o evolutivamente.

**A) Matrices PAM**. Dayhoff (1972) propuso un esquema de puntuación (evolutivo) proporcional a $log(f_{AB}/f'_{AB})$ para el alineamiento del residuo *A* con el *B*, donde $f_{AB}$ es la frecuencia del par *AB* en alineamientos relacionados, y $f'_{AB}$ en los alineamientos no relacionados ($f'_{AB}=f_A x f_B$ el producto de las frecuencias de los residuos en una proteína media). $f_{AB}$ depende de la semejanza entre las proteinas, y para secuencias muy relacionadas los alineamientos de residuos diferentes son poco probables, por lo que $f_{AB}$ es pequeña. A mayores distancias evolutivas, la frecuencia de los reemplazos conservativos crece y las coincidencias entre residuos disminuye. A distancias evolutivas muy grandes las frecuencias se acercan a las que se espera que ocurran al azar, perdiéndose la posibilidad de detectar la señal *biológica*.

Estas *matrices de puntuación* asumen que la estimación de las tasas de mutación para secuencias de proteinas muy relacionadas pueden ser extrapoladas a secuencias menos relacionadas. Para estimar estas tasas, Dayhoff examinó alineamientos múltiples con no más de un 15% de diferencias en comparación por identidad. Si en una posición (columna) de un alineamiento habían 9 alaninas y una serina, entonces se infería que la alanina era un ancestro de la serina en dicha posición. A efectos de calcular una distancia evolutiva se considera que el cambio de alanina a serina ocurría un 10% de las veces. Siguiendo este criterio construyó las tablas de frecuencias para representar las tasas de sustitución entre aminoácidos en secuencias fuertemente relacionadas para esa distancia evolutiva. Esta tabla de frecuencias se normalizó de tal forma que en 100 residuos se observara sólo una tasa de una mutación. Esta frecuencia de mutación aceptada de 1% (*PAM, Percent Accepted Mutation*) se usó para estimar, por extrapolación, matrices a diversas distancias (120, 250, 320, etc) PAMs. La puntuación para cada pareja de residuos se calcula como el logaritmo del cociente de la probabilidad de ocurrencia del par y la probabilidad esperada basada en la frecuencia de cada aminoácido.

La utilización de estos cocientes fue una importante contribución de Dayhoff, y proporcionó un modelo cuantitativo para evaluar la semejanza entre alineamientos. Aunque el conjunto de datos utilizado por Dayhoff era el disponible en 1978, esta limitación ha sido superada actualmente por un procedimiento automático que actualiza el modelo de Dayhoff usando la información disponible en cada momento en las BdeD (Jones et.al. 1992). Es resaltable el hecho de que, a pesar de contar en la actualidad con varios órdenes de magnitud más de datos, no se aprecian diferencias sustanciales en las puntuaciones, a excepción de aquellos aminoácidos raros (triptófano y cisteína) que estaban pobremente representados en la BdeD original que usó Dayhoff. Resultados similares han sido obtenidos en estudios que intentan utilizar secuencias más distantes (Gonnet et al.1992)

**(B) Matrices BLOSUM**. Altschul (1991) observó que, en general, no se conoce *a priori* lo semejante que son las secuencias que deseamos buscar, por lo que es necesario tener mucho cuidado al seleccionar el esquema de puntuación a utilizar. Diversos test empíricos, mostraron que la matriz pam250 resultaba demasiado distante en muchas situaciones y que era mejor utilizar una pam120, o que diferentes alineamientos podían requerir diferentes extrapolaciones del conjunto de datos PAM a matrices pam40, 120 o pam 450, las cuales quedaban justificadas por una mayor sensibilidad en las

tareas de búsqueda.

La hipótesis básica en el modelo de Dayhoff establece que las relaciones lejanas entre secuencias pueden modelarse con suficiente precisión desde la información de secuencias estrechamente relacionadas. Por tanto, la hipótesis favorece fuertemente las posiciones más mutables (aquellas donde se han producido mayor número de sustituciones), mientras que las posiciones más conservadas (menor número de sustituciones) contribuyen a la totalidad de la tabla. Basándose en ello desarrolló una nueva serie de matrices de puntuación a partir de alineamientos de bloques de secuencias. Un *bloque* es una matriz cuyas filas representan segmentos de secuencias protéicas alineadas sin interrupciones (Posfait et.al 1989, Henikoff et.al 1991, 1992). Estas matrices se denominan *BLOSUM*nn (BLOcks SUbstitution Matrices), donde *nn* indica el umbral de identidad utilizado para la seleccionar los bloques de alineamiento. Es interesante observar que un número bajo de blosum (umbral bajo de identidad) equivale o se corresponde con un número alto de PAM (distancia evolutiva grande).

*Nota*: Estos resultados refuerzan la importancia que tienen los esquemas de puntuación que utilicen los programas. El hecho de que un procedimiento no detecte una relación existente entre dos secuencias puede significar que el procedimiento no fue lo suficientemente sensible o selectivo, pero también puede significar que el esquema de pesos usado no era el adecuado.

| | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | B | Z | X | * | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 4 | -1 | -2 | -2 | 0 | -1 | -1 | 0 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 0 | -3 | -2 | 0 | -2 | -1 | 0 | -4 | A |
| R | -1 | 5 | 0 | -2 | -3 | 1 | 0 | -2 | 0 | -3 | -2 | 2 | -1 | -3 | -2 | -1 | -1 | -3 | -2 | -3 | -1 | 0 | -1 | -4 | R |
| N | -2 | 0 | 6 | 1 | -3 | 0 | 0 | 0 | 1 | -3 | -3 | 0 | -2 | -3 | -2 | 1 | 0 | -4 | -2 | -3 | 3 | 0 | -1 | -4 | N |
| D | -2 | -1 | 2 | 6 | -3 | 0 | 2 | -1 | -1 | -3 | -4 | -1 | -3 | -3 | -1 | 0 | -1 | -4 | -3 | -3 | 4 | 1 | -1 | -4 | D |
| C | -4 | -4 | -4 | -5 | 12 | -3 | -5 | -2 | -3 | -1 | -2 | -3 | -1 | -2 | -3 | -1 | -1 | -2 | -2 | -1 | -3 | -3 | -2 | -4 | C |
| Q | -1 | 1 | 0 | 0 | -3 | 5 | 2 | -2 | 0 | -3 | -2 | 1 | 0 | -3 | -1 | 0 | -1 | -2 | -1 | -2 | 0 | 3 | -1 | -4 | Q |
| E | -1 | 0 | 0 | 2 | -4 | 2 | 5 | -2 | 0 | -3 | -3 | 1 | -2 | -3 | -1 | 0 | -1 | -3 | -2 | -2 | 1 | 4 | -1 | -4 | E |
| G | 0 | -2 | 0 | -1 | -3 | -2 | -2 | 6 | -2 | -4 | -4 | -2 | -3 | -3 | -2 | 0 | -2 | -2 | -3 | -3 | -1 | -2 | -1 | -4 | G |
| H | -2 | 0 | 1 | -1 | -3 | 0 | 0 | -2 | 8 | -3 | -3 | -1 | -2 | -1 | -2 | -1 | -2 | -2 | 2 | -3 | 0 | 0 | -1 | -4 | H |
| I | -1 | -3 | -3 | -3 | -1 | -3 | -3 | -4 | -3 | 4 | 2 | -3 | 1 | 0 | -3 | -2 | -1 | -3 | -1 | 3 | -3 | -3 | -1 | -4 | I |
| L | -2 | -2 | -3 | -4 | -1 | -2 | -3 | -4 | -3 | 2 | 4 | -2 | 2 | 0 | -3 | -2 | -1 | -2 | -1 | 1 | -4 | -3 | -1 | -4 | L |
| K | -1 | 2 | 1 | 0 | -3 | 1 | 1 | -2 | -1 | 3 | 2 | 5 | -1 | -3 | -1 | 0 | -1 | -3 | -2 | -2 | 0 | 1 | -1 | -4 | K |
| M | -1 | -1 | -2 | -3 | -1 | 0 | -2 | -3 | -2 | 1 | 2 | -1 | 5 | 0 | -2 | -1 | -1 | -1 | -1 | 1 | -3 | -1 | -1 | -4 | M |
| F | -2 | -3 | -3 | -3 | -2 | -3 | -3 | -3 | -1 | 0 | 0 | -3 | 0 | 6 | -4 | -2 | -2 | 1 | 3 | -1 | -3 | -3 | -1 | -4 | F |
| P | -1 | -2 | -2 | -1 | -3 | -1 | -1 | -2 | -2 | -3 | -3 | -1 | -2 | -4 | 7 | -1 | -1 | -4 | -3 | -2 | -2 | -1 | -2 | -4 | P |
| S | 1 | -1 | 1 | 0 | -1 | 0 | 0 | 0 | -1 | -2 | -2 | 0 | -1 | -2 | -1 | 4 | 1 | -3 | -2 | -2 | 0 | 0 | 0 | -4 | S |
| T | 0 | -1 | 0 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 5 | -2 | -2 | 0 | -1 | -1 | 0 | -4 | T |
| W | -3 | -3 | -4 | -4 | -2 | -2 | -3 | -2 | -2 | -3 | -2 | -3 | -1 | 1 | -4 | -3 | -2 | 11 | 2 | -3 | -4 | -3 | -2 | -4 | W |
| Y | -2 | -2 | -2 | -3 | -2 | -1 | -2 | -3 | 2 | -1 | -1 | -2 | -1 | 3 | -3 | -2 | -2 | 2 | 7 | -1 | -3 | -2 | -1 | -4 | Y |
| V | 0 | -3 | -3 | -3 | -1 | -2 | -2 | -3 | -3 | 3 | 1 | -2 | 1 | -1 | -2 | -2 | 0 | -3 | -1 | 4 | -3 | -2 | -1 | -4 | V |
| B | -2 | -1 | 3 | 4 | -3 | 0 | 1 | -1 | 0 | -3 | -4 | 0 | -3 | -3 | -2 | 0 | -1 | -4 | -3 | -3 | 4 | 1 | -1 | -4 | B |
| Z | -1 | 0 | 0 | 1 | -3 | 3 | 4 | -2 | 0 | -3 | -3 | 1 | -1 | -3 | -1 | 0 | -1 | -3 | -2 | -2 | 1 | 4 | -1 | -4 | Z |
| X | 0 | -1 | -1 | -1 | -2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -2 | 0 | 0 | -2 | -1 | -1 | -1 | -1 | -1 | -4 | X |
| * | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | 1 | * |

Matriz **pam250** (*inferior izquierda*) y **BLOSUM62** (*superior derecha*) para proteinas

Se pueden establecer otras distancias, diferentes de la evolutiva, para comparar secuencias. Por ejemplo, a continuación se muestra una distancia hidrofóbica (determinante en la conservación).

El *valor esperado* (Expected Value) de una matriz representa la puntuación media asignada a una pareja de residuos, y es la sumatoria de los productos de los pesos por las frecuencias de los residuos (frecuencia esperada de aparición de la pareja): $\mathcal{E}_v = \Sigma_{i=1...n} \Sigma_{j=1...n} w(a_i,a_j) * f_i f_j$, siendo: $(a_i,a_j) \in \mathcal{A}^2$,los símbolos del alfabeto de longitud *n*; $w(a_i,a_j)$ el esquema de pesos entre dichos símbolos y $f_i$ y $f_j$, las frecuencias esperadas de aparición de los símbolos.

Los esquemas de peso usados para la comparación de secuencias requieren tener un valor esperado *negativo*, en caso contrario las secuencias tenderían a alinearse a lo largo de toda su longitud. Cuanto mas negativo es el valor esprado, menor nivel de semejanza se requiere para considerar un alineamiento como significativo (por ello el uso de un valor esperado más negativo se asocia con búsqueda de fragmentos mas alejados evolutivamente y se espera encontrar únicamente señales muy robustas...el numero de fragmentos es menor y sus niveles de semejanza son menores)

| | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | B | Z | X |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 17 | | | | | | | | | | | | | | | | | | | | | | |
| R | -3 | 17 | | | | | | | | | | | | | | | | | | | | | |
| N | 8 | 7 | 17 | | | | | | | | | | | | | | | | | | | | |
| D | 1 | 13 | 11 | 17 | | | | | | | | | | | | | | | | | | | |
| C | 17 | -3 | 7 | 1 | 17 | | | | | | | | | | | | | | | | | | |
| Q | 8 | 6 | 16 | 10 | 8 | 17 | | | | | | | | | | | | | | | | | |
| E | 3 | 11 | 13 | 16 | 3 | 11 | 17 | | | | | | | | | | | | | | | | |
| G | 16 | -2 | 8 | 3 | 16 | 10 | 4 | 17 | | | | | | | | | | | | | | | |
| H | 10 | 4 | 14 | 8 | 10 | 16 | 10 | 11 | 17 | | | | | | | | | | | | | | |
| I | 16 | -5 | 6 | 0 | 16 | 7 | 1 | 14 | 8 | 17 | | | | | | | | | | | | | |
| L | 16 | -5 | 6 | 0 | 16 | 7 | 1 | 14 | 8 | 17 | 17 | | | | | | | | | | | | |
| K | 3 | 13 | 11 | 17 | 1 | 10 | 16 | 3 | 8 | 0 | 0 | 17 | | | | | | | | | | | |
| M | 14 | -6 | 6 | -2 | 16 | 7 | 0 | 14 | 8 | 17 | 16 | 0 | 17 | | | | | | | | | | |
| F | 14 | -6 | 6 | -2 | 14 | 6 | 0 | 13 | 7 | 16 | 16 | 0 | 17 | 17 | | | | | | | | | |
| P | 14 | 0 | 10 | 4 | 14 | 11 | 6 | 16 | 13 | 13 | 13 | 4 | 11 | 11 | 17 | | | | | | | | |
| S | 16 | -2 | 10 | 3 | 16 | 10 | 4 | 17 | 11 | 14 | 14 | 4 | 13 | 13 | 16 | 17 | | | | | | | |
| T | 17 | -2 | 8 | 3 | 16 | 10 | 4 | 17 | 11 | 14 | 14 | 3 | 14 | 14 | 16 | 16 | 17 | | | | | | |
| W | 17 | -3 | 7 | 1 | 17 | 8 | 3 | 16 | 10 | 16 | 16 | 1 | 16 | 14 | 14 | 16 | 16 | 17 | | | | | |
| Y | 14 | 0 | 11 | 6 | 13 | 13 | 7 | 14 | 14 | 11 | 13 | 6 | 11 | 11 | 17 | 16 | 14 | 13 | 17 | | | | |
| V | 16 | -5 | 7 | 0 | 16 | 7 | 1 | 14 | 8 | 17 | 17 | 1 | 16 | 16 | 13 | 14 | 16 | 16 | 13 | 17 | | | |
| B | 5 | 9 | 14 | 14 | 4 | 13 | 15 | 6 | 11 | 2 | 3 | 14 | 2 | 2 | 7 | 6 | 5 | 4 | 8 | 3 | 17 | | |
| Z | 7 | 7 | 16 | 11 | 7 | 15 | 13 | 8 | 14 | 5 | 6 | 12 | 5 | 4 | 10 | 9 | 8 | 7 | 11 | 6 | 14 | 17 | |
| X | 17 | -3 | 8 | 1 | 16 | 9 | 3 | 16 | 10 | 15 | 15 | 2 | 14 | 14 | 14 | 16 | 16 | 17 | 14 | 16 | 5 | 7 | 17 |

Matriz de Hidrofobicidad (Engleman & Steitz) escalada de [0,16] a [-6,17]
(http://www.msicam.co.uk/support/insight/hydro.html)

| Fichero | swiss32 | swiss36 | AA | swiss32 | swiss36 | Matriz de Scores | Valor Esperado |
|---|---|---|---|---|---|---|---|
| N.secs | 49340 | 74019 | | | | | |
| L.Min | 3 | 3 | | | | | |
| L.Max | 5217 | 5217 | | | | | |
| Tot.aa | 17380858 | 26840295 | | | | | |
| L.Prom | 352 | 362 | | | | pam 10 | -8.156775 |
| | | | | | | pam 50 | -3.593013 |
| [A] | 7.5865 | 7.5858 | [M] | 2.3668 | 2.3702 | pam 100 | -2.844400 |
| [C] | 1.7024 | 1.6732 | [N] | 4.5240 | 4.4751 | blosum 90 | -1.672698 |
| [D] | 5.3052 | 5.2876 | [P] | 4.9300 | 4.9174 | pam 150 | -1.193298 |
| [E] | 6.3174 | 6.3518 | [Q] | 4.0380 | 3.9942 | pam 200 | -1.146656 |
| [F] | 4.0401 | 4.0898 | [R] | 5.1788 | 5.1475 | blosum 62 | -0.946783 |
| [G] | 6.8896 | 6.8362 | [S] | 7.1768 | 7.1523 | pam 250f | -0.811955 |
| [H] | 2.2341 | 2.2419 | [T] | 5.7698 | 5.6959 | pam 250b | -0.790956 |
| [I] | 5.7035 | 5.8007 | [V] | 6.5277 | 6.5693 | pam 300 | -0.759875 |
| [K] | 5.9156 | 5.9387 | [W] | 1.2650 | 1.2428 | blosum 30 | -0.490475 |
| [L] | 9.2922 | 9.4232 | [Y] | 3.2118 | 3.1846 | pam 450 | -0.399363 |

En la tabla se presentan las frecuencias de los aa obtenidas para la swissprot v.32 y 36. A la derecha los valores esperados para varias matrices de pesos usadas frecuentemente en la comparación de secuencias.

# Comparación por Semejanza

| | G | T | C | C | G | A | C | T | A | G | T | G | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | -3 | 1 | 4 | 4 | -3 | -3 | 4 | 1 | -3 | -3 | 1 | -3 | |
| A | 1 | -3 | -3 | -3 | 1 | 4 | -3 | -3 | 4 | 1 | -3 | 1 | **-3** |
| T | -3 | 4 | 1 | 1 | -3 | -3 | 1 | 4 | -3 | -3 | 4 | -3 | **2** |
| C | -3 | 1 | 4 | 4 | -3 | -3 | 4 | 1 | -3 | -3 | 1 | -3 | **-9** |
| G | 4 | -3 | -3 | -3 | 4 | 1 | -3 | -3 | 1 | 4 | -3 | 4 | **-1** |
| G | 4 | -3 | -3 | -3 | 4 | 1 | -3 | -3 | 1 | 4 | -3 | 4 | **7** |
| A | 1 | -3 | -3 | -3 | 1 | 4 | -3 | -3 | 4 | 1 | -3 | 1 | **-4** |
| G | 4 | -3 | -3 | -3 | 4 | 1 | -3 | -3 | 1 | 4 | -3 | 4 | **1** |
| C | -3 | 1 | 4 | 4 | -3 | -3 | 4 | 1 | -3 | -3 | 1 | -3 | **9** |
| T | -3 | 4 | 1 | 1 | -3 | -3 | 1 | 4 | -3 | -3 | 4 | -3 | **-1** |
| G | 4 | -3 | -3 | -3 | 4 | 1 | -3 | -3 | 1 | 4 | -3 | 4 | **-5** |
| | | **4** | **-6** | **-2** | **3** | **7** | **0** | **-14** | **-13** | **3** | **13** | **-11** | **-4** |

Esquema de pesos utilizado:
[4 puntos] Residuos iguales
[1 punto] Residuos del mismo tipo
purinas (AG), pirimidinas (CT)
[-3 puntos] Combinaciones restantes.
*Nota*: Si se considera la misma frecuencia para cada residuo, el valor esperado de esta matriz sería:

$$E = -0{,}26 = 0{,}25 \times 0{,}25\ [\ 4 \times 4 + 1 \times 4 + (-3) \times 8\ ]$$

| | A | C | G | T |
|---|---|---|---|---|
| A | 4 | -3 | 1 | -3 |
| C | -3 | 4 | -3 | 1 |
| G | 1 | -3 | 4 | -3 |
| T | -3 | 1 | -3 | 4 |

Cálculo de la *semejanza* entre dos secuencias utilizando un esquema de pesos entre los residuos. En cada celda de la matriz se ha anotado el puntaje asignado a la pareja de residuos que forman la celda. En la última fila y columna se indican las sumas de cada diagonal, siendo 13 el valor máximo.

```
Peso por pareja              1  4  4 -3  4  4 -3  1 -3  4
Secuencia horizontal         G  T  C  C  G  A  C  T  A  G  T  G
Secuencia vertical        C  A  T  C  G  G  A  G  C  T  G
Puntaje acumulado            1  5  9  6 10 14 11 12  9 13
```

Cuando la valoración (como en este caso) se realiza desde el inicio de una de las secuencias hasta el final de una de las secuencias se denomina **semejanza *Global***

*Nota*: Obsérvese que el método empleado es **exhaustivo** ya que analiza todas las diagonales posibles.

# Comparación por Semejanza [ LOCAL ]

La *semejanza* ***local***, se calcula comenzando en una celda positiva y se avanza acumulando valores, siempre que el acumulado se mantenga positivo. Si la diagonal termina, o el acumulado cae por debajo de cero, la semejanza local máxima se encuentra desde el inicio hasta el valor máximo acumulado.

Observe que en un alineamiento *global* pueden haber grandes trozos de baja, o incluso negativa, semejanza que no contribuyen a incrementar la puntuación. En el alineamiento local, las zonas no-conservadas normalmente no son incorporadas al alineamiento.

| | G | T | C | C | G | A | C | T | A | G | T | G |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | | (1) | (4) | (4 | | | (4) | (1 | | | (1 | |
| A | (1 | | | | 1 | (4 | | | 4 | (1 | | 1)=2 |
| T | | 4 | (1 | (1) | | -3 | 1 | (4 | | -3 | 4)=5 | |
| C | | (1) | 4 | 4 | | | 4)=6 | 1 | -3 | | 1 | |
| G | (4) | | | -3 | 4 | (1) | | | 1 | 4)=5 | | 4)=7 |
| G | (4) | | | | 4 | 1)=10 | | | (1 | 4 | | (4) |
| A | (1) | | | | (1 | 4)=14 | | | (4 | 1)=2 | -3 | (1) |
| G | (4 | | | | (4) | 1 | | | (1) | 4 | | 4)=12 |
| C | | 1 | (4 | (4) | | | 4 | (1) | | | 1)=9 | |
| T | | (4) | 1)=6 | 1 | | | (1) | 4 | | | (4 | |
| G | (4) | | | | 4)=9 | (1) | | | 1)=11 | (4) | | 4)8 |

Para el mismo ejemplo anterior, se indican únicamente aquellas celdas que han quedado formando parte de algún fragmento o *semejanza local* entre las dos secuencias. Para cada fragmento se indica el inicio con un "abre-paréntesis" y el final con un "cierra-paréntesis", incluyendo el *puntaje* del fragmento (para los fragmentos de longitud mayor que uno)

```
Peso por pareja                1  4  4 -3  4  4 -3  1 -3  4
Secuencia horizontal           G  T  C  C  G  A  C  T  A  G  T  G
Secuencia vertical          C  A  T  C  G  G  A  G  C  T  G
Puntaje acumulado (global)     1  5  9  6 10 14 11 12  9 13
Puntaje acumulado (local)      1  5  9  6 10 14     1     4
Semejanza local (3)           |<--------------->|   <->   <->
```

*Notas*:

- Aunque se trata de un ejercicio académico, éste contiene todos los elementos de cualquier caso real. Los fragmentos sólo incluyen zonas no-conservadas (de baja semejanza) si han acumulado puntaje suficiente para poderla atravezar hasta otra zona conservada que le compense la disminución del puntaje.

- El método descrito sigue siendo *exhaustivo* ya que examina todos los posibles fragmentos.

# Una Heurística para Comparar por Semejanza

Una heurística es un procedimiento que permite hallar una solución aproximada a un determinado problema, sin asegurar que el resultado es óptimo, pero ofreciendo a cambio mayor velocidad de cálculo (y en ocasiones, la única posibilidad de hallar una solución).

El método de comparación por semejanza descrito en el apartado anterior realiza una búsqueda exhaustiva de todos los posibles fragmentos (que empiezan con un puntaje positivo) recorriendo una a una todas las diagonales, y por ello requieren un tiempo proporcional a O(NM) siendo N y M las longitudes de las secuencias en comparación. Para reducir el espacio de búsqueda (y por tanto el tiempo de computación) se intenta localizar las diagonales donde es más probable que se encuentre el mejor fragmento. Una hipótesis tentativa podría ser que las mejores semejanzas se podrían encontrar en aquellas diagonales que mayor número de residuos iguales presenten (identidades). Este problema ya lo hemos resuelto previamente (ver tablas de dispersión) en tiempo proporcional a la suma de las longitudes de las secuencias.

```
Posic: 123456789012                    Diag. 0: 2
Seq.X: GTCCGACTAGTG                    D.   -1: 5   D.1: 2
                                            -2: 2   D.2: 3
Tabla       A :  6, 9                       -3: 1   D.3: 2
Hash        C :  3, 4, 7                    -4: 1   D.4: 3
            G :  1, 5,10,12                 -5: 2   D.5: 2
            T :  2, 8,11                    -6: 2   D.6: 2
                                            -7: 1   D.7: 2
       Secuencia a comparar:                -8: 1   D.8: 1
                                            -9: 0   D.9: 0
Posic: 12345678901                         -10: 1   D10: 0
Seq.Y: CATCGGAGCTG                                  D11: 0
```

En el recuadro se muestra la determinación de las mejores *diagonales por identidad*, utilizando una tabla de dispersión para los símbolos de la secuencia que se usará como incógnita (Seq.X). El número de la diagonal se obtiene restando las posiciones relativas de cada residuo de la secuencia Y con las posiciones en que aparece dicho residuo en la seq.X (usando la tabla de dispersión para saber cuales son esas posiciones). Existen *n+m-1* diagonales, numeradas en el rango [-10,11], siendo 0 la diagonal principal, las positivas a la derecha y las negativas a la izquierda de la diagonal principal. Luego de realizar el conteo se observa que las diagonales -1, 2 y 4 con 5, 3 y 3 identidades respectivamente son las diagonales con mayor número de identidades.

Ahora podríamos concentrar la búsqueda del mejor *alineamiento* (global o local) sin gaps **únicamente** a lo largo de estas 3 diagonales (o las que se elijan, dado como parámetro), lo que representa una **reducción significativa** del espacio de búsqueda.

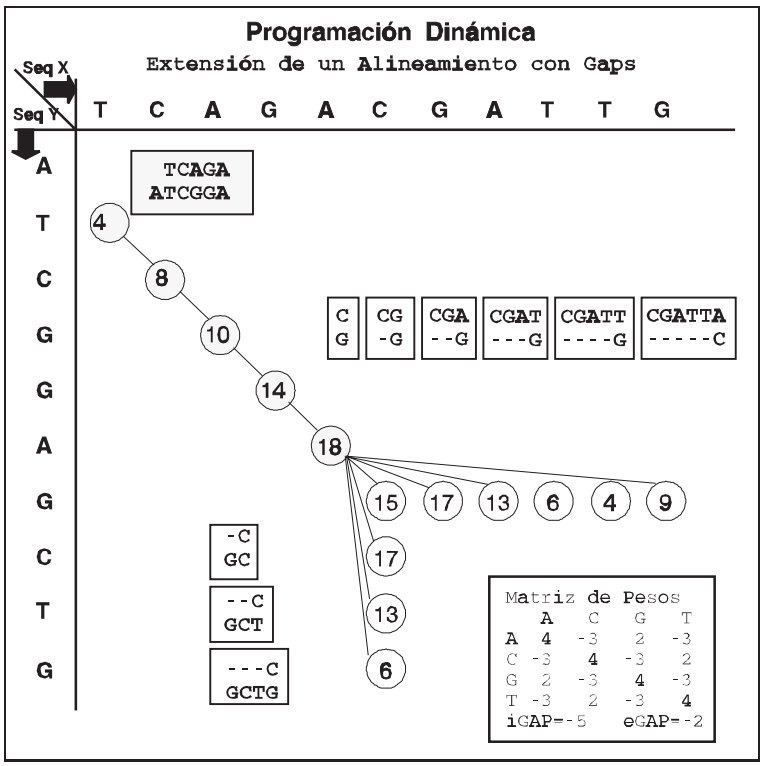
Si bien se puede comprobar para el caso mostrado, como la heurística produce el mismo resultado que el método exhaustivo, con menos cálculos, esto no es válido en todos los casos, y en general, se puede afirmar que siempre existirá un compromiso entre la sensibilidad y selectividad del método (capacidad para incluir todos los positivos y para descartar todos los negativos), contra la velocidad de proceso y en general contra la demanda de recursos computacionales.

# Las Interrupciones [ Gaps ]

Ahora introduzcamos un nuevo *y muy importante,* elemento: las interrupciones y para ello volvamos a los alineamientos por identidad. Siempre es posible incrementar el número de coincidencias introduciendo algunas discontinuidades (*gaps*) en las secuencias. Veamos dos posibilidades:

```
X:  TCAG-ACG-ATTG          X:  TCAGACGATTG
    || | | |  | |              || || | | |
Y: ATC-GGA-GC-T-G          Y: ATCGGA-GCT-G
```

En ambos casos tendríamos una semejanza por identidades del 70%. Pero, ¿ son válidos ambos resultados?, ¿ cómo podemos evaluar sistemáticamente las posibles combinaciones ?. Intuitivamente la idea es simple: supongamos que intentamos extender el fragmento en formación de puntaje 18 que se indica en la figura. El resultado de cada alternativa se muestra en los círculos.




El fragmento en formación (sombreado) {TT, CC, AG, GG, AA} con puntaje 18 puede:

(**a**) Emparejar la siguiente pareja en el fragmento, formada por los residuos C y G de las secuencias X e Y respectivamente. Esta extensión, en la misma diagonal, *no* incluye gaps (pierde 3 puntos, hasta 15).

(**b**) Emparejar el siguiente residuo de la sec.X (residuo C) con alguno de los restantes residuos de la sec.Y (residuos CTC). Esta solución incorpora gaps en la sec.Y

(**c**) Emparejar el siguiente residuo de la sec.Y (residuo G) con alguno de los restantes residuos de la sec.X (residuos GATTG). Esta solución incorpora gaps en la sec.X

Además observemos lo siguiente:

(1) La penalización por los *gaps*, pretende evitar la deformación de las secuencias hasta que pierdan su señal biológica a la búsqueda de incrementar las coincidencias. La forma más empleada penaliza por iniciar (*iG≥0*) y por extender *eG* (*eG≥0*), de la forma: $\alpha_g$=iG+eG(*g*-1), donde *g* es la longitud del gap y $\alpha_g \geq 0$. En general, las funciones no proporcionales (como la lineal descrita), da mejores resultados que las estrictamente proporcionales (Gotoh, 1990).

(2) Los saltos, inclusión de gaps, sólo se pueden dar a la derecha en la siguiente fila y abajo en la siguiente columna de la celda actual. Otras filas u otras columnas indicarían que se han incorporado gaps en *ambas* secuencias (no tiene sentido alinear dos gaps).

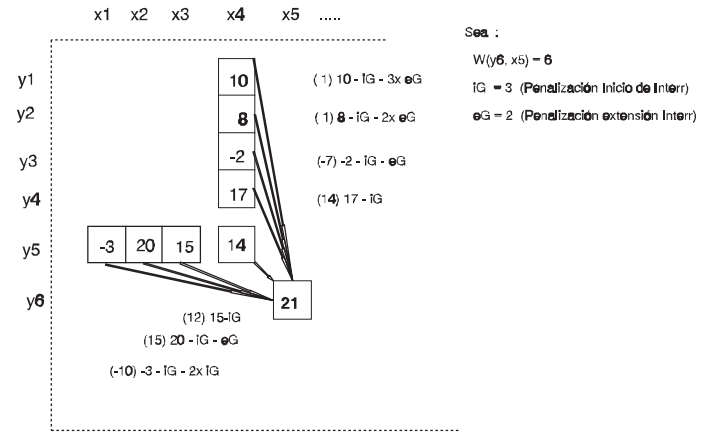
(3) La penalización limita las opciones de salto (mientras el puntaje se mantenga positivo en caso de alineamientos locales), o en su defecto, los programas incorporan alguna estrategia para

limitar el análisis de las opciones de salto, en general basadas en una ventana de inspección. Recuerde, que a más penalización, menos gaps.

(4) Se debe descartar la idea de analizar todos las combinaciones posibles de los elementos de las secuencias con interrupciones. Sin interrupciones, la comparación completa de secuencias de longitud *N* tiene complejidad $N^2$. Para examinar la posibilidad de interrupción en cada posición sería necesario repetir los cálculos *2N* veces, lo que lo hace inviable como tarea rutinaria. La programación dinámica (Needleman & Wunsch, 1970; Sellers, 1974 y 1980, Smith & Waterman; 1981), reduce el espacio computacional con un buen compromiso de sensibilidad.

(5) A efectos de claridad el algoritmo de extensión con gaps de un fragmento se ha mostrado indicando los posibles saltos que se pueden dar desde una celda para llegar a otra. Sin embargo, a efectos de su implementación computacional se suele plantear a la inversa (desde que celdas puedo llegar a la celda actual).

Para ello se construye una matriz *S,* formada por todas las posibles parejas de residuos que pueden obtenerse con las secuencias *X* e *Y* a alinear y se completa con los valores de cada pareja (usando el esquema de pesos *w* para cada par de residuos).



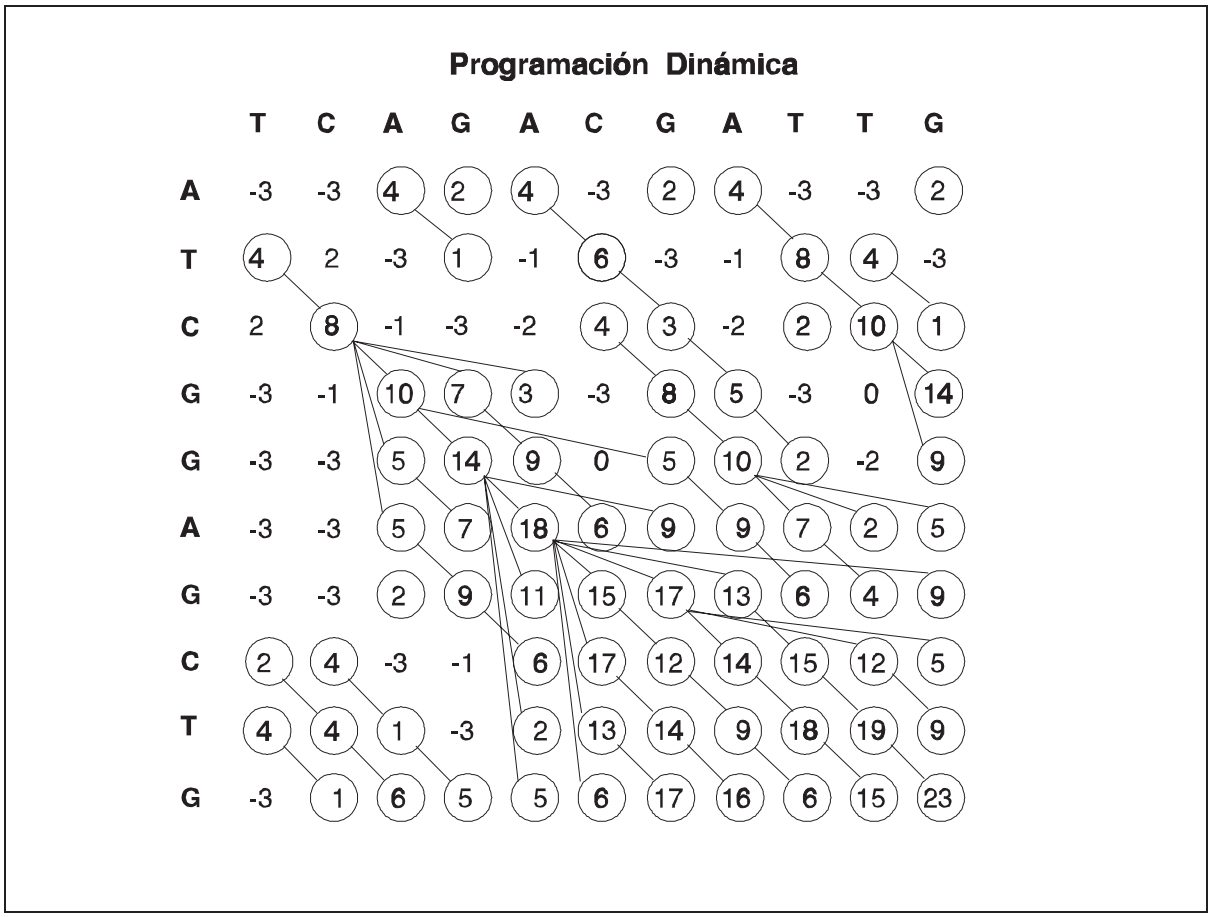

**Evaluación de una celda usando Programación Dinámica**

Además se dispone de las penalizaciones por inicio (iG) y extensión (eG) de gaps. Para calcular una celda (en la figura la celda 6,5) a partir de la matriz *S*, se elige el valor máximo de entre : la celda anterior en la diagonal (5,4 = 14) y los valores de la fila y columna anterior a la izquierda y por encima de la celda, penalizados por iniciar y extender una discontinuidad. En este caso el valor máximo corresponde a la celda (5,2) con valor inicial 20 y valor proyectado 15 (20 -iG -eG = 20 - 3 - 2). Obsérvese que el valor de la celda (4,4) es 17 y penalizado es 14. Si hubiera sido necesario elegir entre este valor y la diagonal, se elegiría la diagonal, que significa que no hay interrupciones. De esta forma cada celda $S_{i,j}$ queda definida a partir de los valores de $S_{i-1,j-1}$, $S_{i-1,j}$ y $S_{i,j-1}$ de acuerdo a la siguiente expresión:

$$S_{i,j} = \max \{ \; S_{i-1,j-1} + w(x_i,y_j), \; S_{i-1,j} + \alpha_x, \; S_{i,j-1} + \alpha_x \}$$

donde $\alpha_x$ es un valor negativo que corresponde a la penalización por iniciar o extender una discontinuidad

Cuando se trata de una *alineamiento local*, adicionalmente se debe hacer $S_{i,j} = \max (S_{i,j}, 0)$.




A continuación se muestra el ejercicio completo de alineamiento de dos secuencias, utilizando **Programación dinámica**. Se ha utilizado los siguientes pesos entre bases : 4 para bases iguales, 2 para bases del mismo tipo (purinas, pirimidinas) y -3 para el resto. Penalización de inicio de interrupción -5 y -2 por extensión de la misma. El valor en cada celda se determina como el máximo de entre los valores de la fila y columna anterior por encima y a la izquierda de la celda (penalizando por las interrupciones) y la celda en diagonal. De esta forma el valor 7 (en la 4ª celda de la diagonal principal) prefiere el valor 8 de la fila anterior (penalizado en -5) al valor de la celda en la misma diagonal. Cuando se inserta más de una interrupción (por ejemplo en la celda (7,8) con valor final 13) se penaliza por inicio y por extensión de la interrupción. De esta forma se obtiene el mejor alineamiento considerando interrupciones, y tendría la forma:

```
 TCAGACGATTG
 ||.||  ..||
ATCGGA--GCTG
```

# Búsquedas Rápidas: FASTA

Los métodos basados en Programación Dinámica requieren un tiempo proporcional a $O(N^2)$. En el caso de búsquedas en bases de datos, el tiempo de computación requerido para este tipo de algoritmos es elevado. Por esta razón, existen otros desarrollos de tipo más heurístico que buscan velocidad a cambio de sensibilidad y selectividad. Para reducir el espacio de búsqueda (y por tanto el tiempo de computación) se intenta localizar las posiciones donde es más probable que se encuentre el mejor alineamiento (posición de mayor coincidencia entre las secuencias). Por ejemplo, la serie de programas FASTA (Lipman & Pearson, 1985, Pearson & Lipman 1988) basa su estrategia en identificar diagonales con mayor número de identidades (residuos iguales) sin incluir interrupciones, lo que le permite realizar una revisión completa de las dos secuencias en un tiempo proporcional a O(N+M), aunque posteriormente reevalúa estas regiones usando matrices de pesos entre residuos para permitir que los reemplazos conservativos también sean tomados en cuenta en la evaluación. Con ello, FASTA logra reducciones de hasta 2 órdenes de magnitud en los tiempos de búsqueda.

La idea se basa en realizar una primera etapa de evaluación de la semejanza entre dos secuencias, considerando únicamente coincidencias y sin permitir interrupciones (tal como hemos visto con anterioridad). Si se utiliza este criterio entonces es posible recurrir a una tabla de dispersión (hash) que contenga las posiciones de cada residuo (o grupo de residuos si se utilizara una *k-tupla* > 1) de la secuencia que se está analizando (para su construcción se requiere recorrer una sola vez dicha secuencia). Con este procedimiento se detecta la diagonal en que se produce una coincidencia para cada residuo de la secuencia de comparación, acumulando para obtener las mejores diagonales (normalmente 10). Este proceso requiere un tiempo proporcional a la longitud de la segunda secuencia. De esta forma el tiempo para comparar dos secuencias, usando FASTA, es proporcional a la suma de las longitudes de las secuencias, frente a un tiempo proporcional al producto de las longitudes de las secuencias en los métodos de programación dinámica.

Pasada esta primera etapa de filtro, que ha permitido detectar regiones iniciales con alto grado de coincidencias, se procede a recalcular la semejanza de las regiones, pero esta vez ya utilizando un esquema de puntuaciones tradicional. A partir de estas regiones recalculadas se intenta la unión de regiones mediante la inserción de interrupciones.

Para asegurar resultados equivalentes a los métodos exhaustivos, una vez detectadas las mejores secuencias (las de mayor semejanza a lo largo de la búsqueda), se realiza una última comparación utilizando un esquema de programación dinámica.

Debido a estas características, la recomendación biológica suele ser, realizar primero una búsqueda con estos algoritmos rápidos, y luego una comparación exhaustiva sobre las mejores secuencias empleando los algoritmos de programación dinámica.

Nota: En el documento "Flexible sequence similarity searching with the FASTA3 program package", William R. Pearson (http://www.people.virginia.edu/~wrp) realiza una revisión actualizada de la serie de programas FASTA. El dodumento incluye la descripción completa de los diversos programas que hoy en día forman el paquete FASTA, el análisis de las opciones de cada uno, la compración con otros métodos (especialmente de la serie BLAST), así como recomendaciones acerca de como valorar los resultados y consejos acerca de que opciones y parámetros usar en cada caso.

# *Basic Local Alignment Search Tool*

Altschul, S.F., Gish W., Miller W., Myers E.W., and Lipman D.J.
J. Mol. Biol. (1990) 215:403-10.

Las limitaciones de la identificación de las mejores diagonales utilizando criterios de identidad propuestos por FASTA tratan de ser superadas por la serie de programas BLAST.

En pocas palabras, de la misma forma que FASTA busca *k-tuplas* iguales (palabras de longitud k) para identificar diagonales potencialmente interesantes (perdiendo información de los reemplazos conservativos y de palabras más pequeñas que k), BLAST amplía la búsqueda a todos aquellas palabras que producen emparejamientos positivos. Puesto que ello amplía el espacio de búsqueda, una forma de volverlo a reducir es descartando aquellos emparejamientos que estadísticamente no sean probables de ocurrir. Para ello es necesario tener en cuenta dos aspectos básicos:

(a) La identificación rápida de segmentos, y de entre ellos, del *maximal segment pair (MSP)*. Un segmento es una subsecuencia continua de cualquier longitud. El puntaje de dos segmentos alineados de la misma longitud se obtiene sumando de los valores de semejanza de cada pareja de residuos del alineamiento (matriz de semejanza entre residuos).

(b) La elección (para su análisis en detalle) de aquellos segmentos que sean más probables de estar contenidos en el alineamiento final.

Para resolver (a) procede a identificar todas las palabras de longitud $w$ de la secuencia incógnita ($S_q$) que puntúen positivamente con cada palabra de la secuencia de la base de datos ($S_{DB}$) con la que está siendo comparada (obsérvese que este proceso es muy parecido a la identificación de *k-tuplas* que hace FASTA, aunque FASTA trabaja sobre identidades).

Por ejemplo, para $w=4$ tendremos que generar (inicialmente) una lista de $20^4$ (160.000) entradas que contendrá las posiciones de todas las palabras (de longitud 4) de $S_q$ que puntúen positivamente. La primera entrada corresponderá a la palabra AAAA, con la cual se recorrerá $S_q$ buscando palabras de tamaño 4 que puntuación positiva (usando una matriz de semejanzas), y se anotará como dato la posición de inicio. Si se programa cuidadosamente, esta lista puede ser elaborada en tiempo proporcional a la longitud de la misma y necesitar bastante menos entradas de las previstas.

Para la parte (b) se intenta reducir la lista anterior dejando únicamente aquellas palabras que puntúen por encima de un umbral $T$. Los valores de $T$ pueden estimarse a partir de los resultados de Karlin & Altschul (1990) que estiman matemáticamente el puntaje por debajo del cual es probable que un segmento sea aleatorio.

Como se puede observar la longitud de la lista (lo cual se reflejará en el tiempo de cálculo) es dependiente de los valores que se asignen a $w$ y $T$.

Supongamos que se desea construir la lista inicial de dispersión de BLAST, para una secuencia de ADN (alfabeto ACGT), cuya matriz de semejanzas por residuos se muestra:

```
   A  C  G  T
A  4 -3  1 -3
C -3  4 -3  1
G  1 -3  4 -3
T -3  1 -3  1
```

Sea $S_q$ = { TCAGACGATTGAAC } (L=14)

Si proponemos *w=2* (académicamente), tendremos 16 ($4^2$) posibles palabras { AA, AC, AG, AT, CA, CC, CG, CT, GA, GC, GC, GT, TA, TC, TG y TT }. El problema consiste en buscar cada una de estas palabras en $S_q$ anotando su posición y puntaje:

```
T>=1 ----------------------------------------
AA [ 9]  (CA, 2, 1)(AG, 3, 5)(GA, 4, 5)(AC, 5, 1)(GA, 7, 5)(AT, 8, 1)(GA,11, 5)(AA,12, 8)(AC,13, 1)
AC [ 6]  (TC, 1, 1)(AG, 3, 1)(AC, 5, 8)(AT, 8, 5)(AA,12, 1)(AC,13, 8)
AG [10]  (AG, 3, 8)(GA, 4, 2)(AC, 5, 1)(CG, 6, 1)(GA, 7, 2)(AT, 8, 1)(TG,10, 1)(GA,11, 2)(AA,12, 5)(AC,13, 1)
AT [ 6]  (AG, 3, 1)(AC, 5, 5)(AT, 8, 8)(TT, 9, 1)(AA,12, 1)(AC,13, 5)
CA [ 7]  (CA, 2, 8)(GA, 4, 1)(CG, 6, 5)(GA, 7, 1)(TG,10, 2)(GA,11, 1)(AA,12, 1)
CC [ 6]  (TC, 1, 5)(CA, 2, 1)(AC, 5, 1)(CG, 6, 1)(TT, 9, 2)(AC,13, 1)
CG [ 4]  (CA, 2, 5)(AG, 3, 1)(CG, 6, 8)(TG,10, 5)
CT [ 5]  (TC, 1, 2)(CA, 2, 1)(CG, 6, 1)(AT, 8, 1)(TT, 9, 5)
GA [ 6]  (CA, 2, 1)(AG, 3, 2)(GA, 4, 8)(GA, 7, 8)(GA,11, 8)(AA,12, 5)
GC [ 7]  (TC, 1, 1)(GA, 4, 1)(AC, 5, 5)(GA, 7, 1)(AT, 8, 2)(GA,11, 1)(AC,13, 5)
GG [ 7]  (AG, 3, 5)(GA, 4, 5)(CG, 6, 1)(GA, 7, 5)(TG,10, 1)(GA,11, 5)(AA,12, 2)
GT [ 7]  (GA, 4, 1)(AC, 5, 2)(GA, 7, 1)(AT, 8, 5)(TT, 9, 1)(GA,11, 1)(AC,13, 2)
TA [ 9]  (TC, 1, 1)(CA, 2, 5)(GA, 4, 1)(CG, 6, 2)(GA, 7, 1)(TT, 9, 1)(TG,10, 5)(GA,11, 1)(AA,12, 1)
TC [ 5]  (TC, 1, 8)(AC, 5, 1)(TT, 9, 5)(TG,10, 1)(AC,13, 1)
TG [ 6]  (TC, 1, 1)(CA, 2, 2)(AG, 3, 1)(CG, 6, 5)(TT, 9, 1)(TG,10, 8)
TT [ 4]  (TC, 1, 5)(AT, 8, 1)(TT, 9, 8)(TG,10, 1)

T>=5 ----------------------------------------------
AA [ 5]  (AG, 3, 5)(GA, 4, 5)(GA, 7, 5)(GA,11, 5)(AA,12, 8)
AC [ 3]  (AC, 5, 8)(AT, 8, 5)(AC,13, 8)
AG [ 2]  (AG, 3, 8)(AA,12, 5)
AT [ 3]  (AC, 5, 5)(AT, 8, 8)(AC,13, 5)
CA [ 2]  (CA, 2, 8)(CG, 6, 5)
CC [ 1]  (TC, 1, 5)
CG [ 3]  (CA, 2, 5)(CG, 6, 8)(TG,10, 5)
CT [ 1]  (TT, 9, 5)
GA [ 4]  (GA, 4, 8)(GA, 7, 8)(GA,11, 8)(AA,12, 5)
GC [ 2]  (AC, 5, 5)(AC,13, 5)
GG [ 4]  (AG, 3, 5)(GA, 4, 5)(GA, 7, 5)(GA,11, 5)
GT [ 1]  (AT, 8, 5)
TA [ 2]  (CA, 2, 5)(TG,10, 5)
TC [ 2]  (TC, 1, 8)(TT, 9, 5)
TG [ 2]  (CG, 6, 5)(TG,10, 8)
TT [ 2]  (TC, 1, 5)(TT, 9, 8)

T>=8 ----------------------------------------------
AA [ 1]  (AA,12, 8)
AC [ 2]  (AC, 5, 8)(AC,13, 8)
AG [ 1]  (AG, 3, 8)
AT [ 1]  (AT, 8, 8)
CA [ 1]  (CA, 2, 8)
CC [ 0]
CG [ 1]  (CG, 6, 8)
CT [ 0]
GA [ 3]  (GA, 4, 8)(GA, 7, 8)(GA,11, 8)
GC [ 0]
GG [ 0]
GT [ 0]
TA [ 0]
TC [ 1]  (TC, 1, 8)
TG [ 1]  (TG,10, 8)
TT [ 1]  (TT, 9, 8)
```

En cada caso (T=1, 5, 8) se indica para cada palabra el número de veces que aparece otra palabra de $S_q$ que satisface el umbral. Cada elemento de la lista está formado por la palabra de $S_q$ con la que se ha producido el emparejamiento, su posición y puntaje (estrictamente sólo es necesaria la posición). Obsérvese cómo al aumentar el umbral disminuye el número de palabras de $S_q$ que lo superan, y por tanto se ganaría en velocidad (menos *extensiones*), siempre a costa de aumentar la probabilidad de perder algún emparejamiento de interés. Supongamos ahora que tenemos una secuencia que queremos comparar. Sea $S_{DB}$= { GAC.... }, y puesto que *w=2*, la primera palabra a buscar sería *GA*. Si usamos T=8, deberíamos extender 3 *hits* { posiciones 4, 7 y 11 }, y si trabajáramos con T=5 tendríamos que extender 4 *hits* (incluyendo AA en la posición 12).

# *Gapped BLAST and PSI-BLAST*
# *A new Generation of Protein DB search Programs*

Altschul, S.F., Madden T.L., Schaffer A.A., Zhanng J., Zhang Z., Miller W., and Lipman D.J.
Nucleid Acids Research (1997) v.25, n.17 3389-3402

Esta nueva serie de programas incorporan tres grandes refinamientos a la versión original:

(1) *Two-hits method*. La propuesta inicial busca palabras con puntajes mayores que un umbral T a partir de las cuales se realizan las extensiones en busca de los mejores segmentos. El proceso de extensión es el más caro computacionalmente. Para incrementar la velocidad ahora se exige la presencia de al menos *dos* palabras que no se solapen, que estén en la misma diagonal y que como mucho se encuentren a una distancia *A* entre ellas. Para recuperar sensitividad, el umbral *T* debe disminuir, aunque a cambio se extienden menos parejas.

(2) *Gapped segments.* Incorpora la posibilidad de generar alineamientos con interrupciones (*gaps*). La versión original encontraba, en general, varios fragmentos en la comparación entre dos secuencias, que considerados en conjunto resultaban significativos. Ahora se analiza si es posible que algunos de estos fragmentos correspondan al mismo alineamiento con gaps, y puesto que ahora el objetivo es "el mejor alineamiento con gaps", entonces el valor de T puede incrementarse (para descartar fragmentos menos significativos), lo cual incrementa la velocidad de la aplicación.

La inclusión de gaps requiere programación dinámica (complejidad NM) para extender la pareja de hits en ambas direcciones. Para recuperar velocidad hay propuestas que reducen la ventana de inspección (máximo tamaño de gaps) lo que produce que la evaluación se centre en una franja alrededor del alineamiento mayor. La propuesta que incorpora esta versión de BLAST considera sólo aquellos alineamientos alternativos que no disminuyen el puntaje más allá de un parámetro $X_g$.

(3) *Búsqueda iterativa* usando una matriz de pesos específica por posición, que se genera en la iteración *i* a partir de los alineamientos significativos que se han encontrado, y que es usada en la iteración *i+1* para intentar recuperar nuevos alineamientos (de forma similar a la búsqueda por motivos).

# Clasificación de las Aplicaciones

Las aplicaciones de comparación de secuencias son casos particulares de la tarea de encontrar las relaciones de semejanza existentes entre el grupo de secuencias { $X_1$, $X_2$, ..., $X_n$ } y las secuencias del grupo { $Y_1$, $Y_2$, ..., $Y_m$ }. Se presentan los siguientes casos :

n=1 m=1 *Comparación de parejas de secuencias (pairwise comparison).*
Este caso corresponde, al estudio de las relaciones entre secuencias revisado hasta ahora y además es una etapa parcial de los alineamientos múltiples de secuencias. Una variante interesante es el alineamiento de clusters de secuencias (grupos de secuencias tratados como una sola), que será revisada en detalle a continuación.

n=1 m>>1 *Búsquedas en Bases de Datos.*
Este caso especial es frecuentemente en biología molecular cuando se compara una secuencia test con el conjunto de las secuencias conocidas almacenadas en las bases de datos. Se trata de encontrar semejanzas que permitan inferir algunas de las propiedades de la secuencia investigada.

La demanda computacional en este tipo de aplicaciones es muy elevada: sobre las 4 horas de CPU para una secuencia típica (400 residuos en proteína o 1200 residuos en ADN), en una estación de trabajo SUN-Sparc 10 (Swissprot, v.30, y EMBL v.40). Otro de los problemas que se presenta en este tipo de aplicaciones es que debido al alto volumen de datos y su crecimiento exponencial, la probabilidad de que se produzcan coincidencias debidas al azar aumenta. Este problema se conoce como el *ruido de fondo* de las búsquedas en BdeD y su efecto es la aparición de lo que se podría denominar *falsos positivos* (referidos a las semejanzas entre secuencias). En esta línea, es necesario el uso de algoritmos más complejos que proporcionen criterios cuantitativos para decidir cuando una relación tiene significado biológico y cuando es más probable que se haya producido al azar.

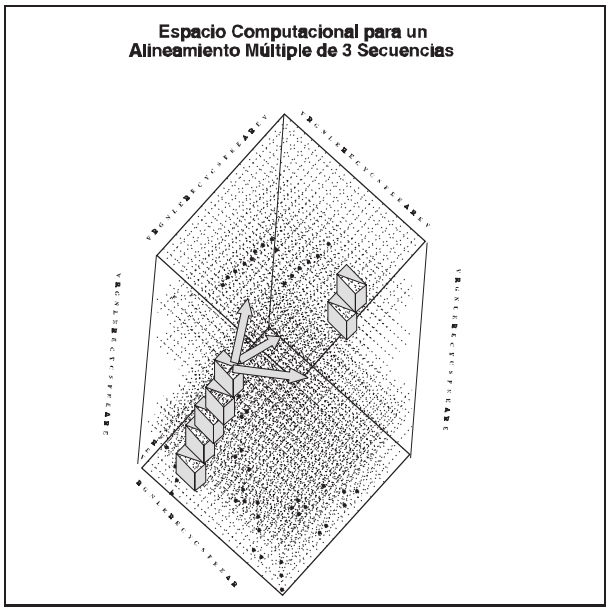
n>1 m>1 *Varias contra varias.*
El caso n>1 y m>>1 es una extensión de la búsqueda en bases de datos y su solución encaja en el grupo anterior. Esta aplicación es cada vez más frecuente, sobre todo cuando se trabaja en paralelo, justamente porque permite incrementar la relación computación/comunicaciones al existir más de una secuencia contra la que comparar un grupo extraído de la base de datos y distribuido a los procesadores.

Otra aplicación importante aparece para $X_i = Y_i$ i=1...n (Todas *versus* Todas). Esta suele ser una de las etapas iniciales de los procesos de alineamiento múltiple de secuencias (determinación de la matriz de semejanzas múltiples) como paso previo a la determinación de la topología del alineamiento, y también en los procesos de clasificación de secuencias en familias y grupos.

Si el problema de la búsqueda en BdeD es ya sumamente interesante computacionalmente cuando se trabaja sobre 40 mil proteinas, en este caso hablamos de medio millón de comparaciones en un alineamiento múltiple de 1000 secuencias.

# Alineamiento Múltiple de Secuencias

Hoy en día, prácticamente todo análisis de secuencias biológicas rutinariamente comienza, contiene o concluye con un alineamiento múltiple (*AM*) de aquellas secuencias que presumiblemente son homólogas a la secuencia problema (Coulson *et al.* 1987; Gonnet *et al.* 1992). Estos alineamientos son también el punto de partida de métodos para predecir la estructura secundaria, para estimar los diferentes tipos de plegamientos de proteinas, para interpretar datos del genoma humano o como pasos intermedios en la construcción de árboles filogenéticos.




El *AM* de secuencias puede ser visto como una generalización del alineamiento de parejas, donde la complejidad de esta aproximación crece exponencialmente con el número de secuencias que intervienen. La posibilidad de resolver el problema por métodos *manuales* (Barber y Maizel, 1990), debe quedar descartada por la enorme complejidad del problema, limitándose su uso a casos con pequeños grupos. Por analogía con la comparación de parejas, es posible aplicar la misma técnica de programación dinámica al caso exhaustivo multidimensional, pero el crecimiento exponencial de la complejidad de método limitó las tempranas implementaciones a no más de tres dimensiones (Murata *et al.* 1985; Gotoh, 1986) (por ejemplo, si alinear dos secuencias de 300 residuos tardase un segundo, alinear 3 tardaría 300s y alinear 10 tardaría sobre los $300^8$s, lo cual es más que el tiempo transcurrido desde el inicio del universo).

Carrillo y Lipman (1988) introdujeron un esquema de acotamiento que utiliza alineamientos de parejas para limitar el volumen del espacio N-dimensional que se necesita para el alineamiento exhaustivo, resultando en importantes mejoras en la velocidad, que posibilitaron su implementación en el programa MSA (Lipman *et al.* 1989) y la aplicación del método a pequeños grupos de secuencias. Por lo general tampoco es posible alinear más de 10 secuencias usando el riguroso modelo del programa MSA. De esta forma es fácilmente entendible la proliferación de métodos aproximados para conjuntos más grandes de secuencias. La solución clásica se basa en la formación de *clusters* de secuencias, los cuales se resuelven progresivamente (Corpet, 1988; Gotoh 1993; Miller 1993). Para ello, dada una medida del parecido o semejanza entre dos secuencias, se eligen aquel par correspondiente al valor más alto, y se alinean y agrupan entre sí para formar un único *cluster* de secuencias. A partir de este momento este cluster será tratado como una sola secuencia, y el proceso se repetirá hasta tener un solo cluster con todas las secuencias que intervenían en el *AM*.

Posiblemente los programas más difundidos de este tipo sean *CLUSTAL-V* (Higgins *et al.* 1992) o CLUSTAL-W en su última versión (Thompson *et al.* 1994), y *PILEUP*, el cual utiliza el método heurístico propuesto por Wilbur y Lipman (1983) para el cálculo de los niveles de semejanza entre las secuencias. La aplicación de esta estrategia no garantiza resultados óptimos (en una primera etapa evalúa por coincidencias de residuos, por lo que si utilizara esquemas de puntuación podrían haber otras soluciones posibles con puntuación mayor). Sin embargo, su *sensibilidad biológica* (capacidad

de poner de manifiesto las relaciones biológicas entre las secuencias) es bastante aceptable, y permiten alinear algunos pocos cientos de secuencias (Barton and Stemberg, 1987; Taylor 1987). En la línea de mejorar la sensibilidad biológica de estos resultados, también se han propuesto mumerosas posibilidades *híbridas*, tales como la representada por el programa MACAW (*Múltiple Alignment Construction and Analysis Workbench*, Schuler *et al.* 1991), basado tanto en información estadística sobre alineamientos de segmentos de las secuencias como en la posibilidad de interacción con el usuario. Así mismo, se ha propuesto la utilización de matrices de consenso (Alexandrov, 1992) para describir la preferencia de cada aminoácido, o de cada interrupción, para posicionarse en el alineamiento. También se ha recurrido a información sobre las estructuras secundarias (Smith and Smith, 1992; Godzik and Skolnick, 1994), o a la localización de motivos y/o dominios conservados en las secuencias (Boguski *et al.* 1992), o a la utilización de las características físico-químicas de los aminoácidos en cada posición (Livingstone and Barton, 1993).

Existen pocas propuestas de soluciones en paralelo para el problema del *AM*. La primera de ellas (Gonnet *et al.* 1992) propone el alineamiento completo de una BdeD de proteinas (MIPS versión 64, con aproximadamente 18 mil secuencias). La solución completa del problema habría necesitado 35 x $10^9$ comparaciones de parejas de secuencias (del orden de $10^6$ años de CPU si consideramos un seg/pareja). Por esta razón el problema se simplifica organizando las secuencias en un árbol de búsqueda ordenado *alfabéticamente* (según la composición de residuos en las secuencias), y se asume que las secuencias similares estarán juntas en dicho árbol, descartando las comparaciones entre parejas lejanas de secuencias. Esto reduce el número de comparaciones a realizar hasta 6.5 x $10^6$, de forma que *sólo* se necesitaron 405 días de CPU (6 estaciones de trabajo ejecutando el programa en *background* durante 19 semanas). La paralelización se realizó a nivel de tareas completas y cada estación de trabajo accedía a la BdeD a través del sistema de ficheros.

La segunda propuesta (Date *et al.* 1993), se refiere al *AM* de pequeños grupos de secuencias usando métodos basados en los esquemas de alineamiento jerárquicos. La paralelización utiliza un esquema maestro-esclavo (*farming approach*) a nivel de tareas, aunque únicamente para la primera etapa de cálculo de la semejanza entre todas las secuencias, puesto que las restantes etapas se resolvieron en el procesador maestro. La primera etapa suele ser ciertamente la más cara computacionalmente, pero en este caso lo es más, puesto que la propuesta incluye el cálculo de lo que denominan *nivel medio de semejanza*, para lo cual realizan para cada par de secuencias 100 alineamientos aleatorios, con lo cual se incrementa drásticamente el peso de la tarea de comparación de todas las secuencias contra todas, o lo que es lo mismo, se minimiza el efecto de la tarea de más difícil paralelización: la solución de la topología del alineamiento. La propuesta no presenta problemas de comunicaciones (secuencias en memoria, y tiempo de planificación mínimo con respecto a la carga de trabajo).

En general, los algoritmos jerárquicos resuelven el alineamiento múltiple de secuencias en tres etapas:

a) Cálculo de la semejanza entre cada par de secuencias que definen la *matriz de semejanza múltiple* $S_{i,j}$. Esta etapa requiere *[N(N+1)/2]* comparaciones para un conjunto *N* de secuencias.
b) Determinación de la topología del alineamiento (el orden en que se realizará el alineamiento). La topología se obtiene en base al nivel de semejanza entre parejas, teniendo prioridad aquellas parejas más semejantes.
c) Resolver el alineamiento. En este caso se resuelven *N-1* alineamientos, siguiendo el orden determinado en la etapa anterior. Conforme se asciende en la topología hacia los nodos superiores, la complejidad del algortimo crece (alineamientos de clusters).

Como se observa, la demanda computacional involucrada en el *AM* es elevada, y en general sólo es

posible realizar este análisis si se trabaja con grupos o familias relativamente pequeñas de secuencias, en los cuales las relaciones o niveles de semejanza entre las secuencias que participan en el *AM* son todos elevados (al tratarse de familias de secuencias todos los miembros tienen estrecha relación, que es justamente lo que define una familia). Si tomamos como ejemplo un *AM* de 1000 secuencias (las proteinas-G, por ejemplo), esto representará más de medio millón de comparaciones de parejas de secuencias (recuérdese que un barrido exhaustivo de una BdeD de proteinas, que requiere del orden de 40 mil de estas comparaciones, se realiza en aproximadamente una hora de CPU en una estación de trabajo SUN Sparc 10). Si extrapoláramos el ejemplo hasta la BdeD completa, estas cifras serían del orden de $8x10^8$ comparaciones, con lo que los tiempos serían intratables, incluso para su resolución en paralelo.

Para finalizar esta introducción, y con el objeto de justificar plenamente la necesidad real de abordar estos problemas computacionales, deberíamos responder explícitamente a la siguiente pregunta: ¿Porqué es de interés realizar estos alineamientos sobre grandes grupos de secuencias y no únicamente sobre secuencias que se sabe están relacionadas?. La primera respuesta que podemos dar es simple : es de esperar que los componentes de las familias aumenten rápidamente y, entonces, el problema de alinear componentes de familias se transformará en un problema de alinear grandes grupos de secuencias. Desde esta perspectiva nuestro planteamiento proporciona una solución *por adelantado* a un problema previsible en un futuro cercano.

Sin embargo, otra razón quizás más importante en la actualidad, es que podemos obtener mucha información de calidad a partir de estos resultados. De forma que este análisis nos permitirá a su vez responder a las preguntas clásicas acerca de ¿cuál es el *coste* de una mutación? y ¿ cuál es el *costo* de una interrupción (inserción o delección de un residuo)?, lo que permitirá afinar los parámetros para realizar análisis de secuencias, en general, y para mejorar los *AM*, en particular.

Hoy en día, las mutaciones en un alineamiento (diferencias en aminoácidos) son puntuadas usando las matrices de puntuación (pam, blosum, etc.). Estas matrices se derivan para conjuntos de secuencias genércios (con marcada semejanza a nivel primario). Se ha demostrado (Altschul, 1991) que existen diferencias importantes cuando las mismas matrices son obtenidas para grupos de secuencias con una relación distante (su nivel de semejanza es suficiente para sugerir que están relacionadas, pero no lo es tanto como para que esta relación sea evidente). La gran cantidad de información disponible actualmente en las BdeD hace previsible que, en general, las familias de secuencias tengan un número apreciable de miembros, de forma que resulte interesante obtener independientemente las matrices de puntuación adecuadas para cada uno de estos conjuntos.

Más aún, la localización de las interrupciones es otra de las grandes dificultades de los *AM*. Las interrupciones suelen ser evaluadas con funciones del tipo ($i$+$ek$), donde $k$ es la longitud del gap, e $i$ y $e$ son penalizaciones por iniciar y extender el gap. No hay justificación teórica ni empírica para este tratamiento y aunque en general funciona, las mayores discusiones se deben a que, en algunos casos, la localización de estas interrupciones es inapropiada, hecho que es crítico para la predicción de como se pliega la secuencia. La tendencia actual es que las penalizaciónes no tienen por qué ser las mismos para diferentes familias y por tanto, desde esta perspectiva podrían utilizarse los *AM* sobre grandes grupos de secuencias para determinar los valores de las penalizaciones por interrupciones más adecuados para cada familia (Gonnet *et al.* 1992, Gotoh, 1993).

# Comentarios Finales

(a) El conocimiento del funcionamiento interno de los algoritmos y los aspectos teóricos que los sustentan es importante para entender y valorar los resultados que se obtienen de la ejecución de los programas. En dependencia de los valores que se utilicen como parámetros (umbrales de corte, penalizaciones, longitud de palabra, matrices de pesos, etc.) se obtendrán unos u otros resultados.

(b) Actualmente existe una amplia galería de software disponible en el dominio público, por lo que es conveniente, antes de pretender codificar un algoritmo, buscar que ofertas existen, ya contrastadas que realizan algo similar o lo mismo. Por otra parte, existen muchos casos particulares que no están contemplados y que exigen un desarrollo propio.

(c) El punto de partida para desarrollar software de análisis de secuencias es saber que es lo que se desea codificar. Una buena pista es resolver el problema a mano y preguntarse si se dispone de una metodología sistemática que pueda ser descrita como una serie de pasos a seguir para hallar una solución. Si la respuesta es afirmativa, seguramente se podrá codificar un programa que realice la tarea más rápido y mas preciso que manualmente.

(d) Debido al gran volumen de datos que se dispone, es frecuente que los algoritmos orientados al análisis de secuencias biológicas sean grandes consumidores de recursos computacionales. Por esta razón, las tareas de optimización (lo que incluye técnicas de programación paralela y uso de multiprocesadores) son mandatorias.

(e) Por ello, el trabajo conjunto de las contrapartes biológica y computacional puede ser tremendamente fructífero.

# Referencias

Alexandrov, N. (1992), "Local multiple alignment by consensus matrix", CABIOS, 8(4), 339-345

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. (1990), "Basic local alignment search tool", J.Mol.Biol, (215), 403-410

Altschul, S.F. (1991) "Amino acid substitution matrices from an information theoretic perspective". J.Mol.Biol, (219), 555-565

Barber, A.M. and Maizel, J.V.J (1990), "Sequence editing-aligner : A multiple sequence editor and aligner", Genet.Anal.Tech.Appl., (7), 39-45

Barton, G.J., and Sternberg, M.J.E. (1987) "Evaluation and improvements in the automatic alignment of protein sequences", Protein Eng., (1), 89-94

Boguski, M.S., Hardison, R.C., Schwarz, S. and Miller, W., (1992), "Analysis of conserved domains and sequence motifs in cellular regulatory proteins and locus control regions using new software tools for multiple alignment and visualization", New Niol., (483), 247-260

Carrillo, H. and Lipman D.J. (1988), "The multiple sequence alignment problem in biology", SIAM J.Appl.Math, (48), 1073-1082

Coulson A.F.W.; Collins, J.F., and Lyall, A., (1987) "Protein and Nucleid Acid sequence database searching: a suitable case for Parallel Processing", Computer J., (39), 420-424

Corpet, Florence (1988), "Multiple sequence alignments with hierarchical clustering", NAR (16), 10881-10890

Date S, Kulkarni,R.; Kulkarni,B.; Kulkarni-kale, U. and Kolaskar,A. (1993), "Multiple alignment of sequences on parallel computers". CABIOS (9)4, 397-402

Dayhoff, M.O. (1972), "Atlas of Protein sequences and structures", National Biomedical Research Foundation, Silver Spring, MD; (5), 3.

Doolittle, R.F. (1981), "Similar amino acid sequences: chance or common ancestry?", Science, (214), 149-159

Godzik, A. and Skolnick, J. (1994), "Flexible algorithm for direct multiple alignment of protein structures and sequences", CABIOS 10(6) 587-596

Gonnet, G.H., Cohen, M.A. and Benner, S.A. (1992) "Exhaustive matching of the entire protein sequence database". Science, (256), 1443-1445

Gotoh, O. (1986), "Alignment of three biological sequences with an efficient traceback procedure", J.Theor.Biol, (121), 327-337

Gotoh, O. (1993), "Optimal alignment between groups of sequences and its application to multiple sequence alignment". CABIOS (9)2, 361-370

Henikoff,S. and Henikoff,J.G. (1991) "Automated assembly of protein blocks for database searching". NAR (19), 6565-6572

Henikoff,S. and Henikoff,J.G., (1992) "Amino acid substitutions matrices from protein blocks", Proc.Natl.Acad.Sci. USA, (89), 10915-10919

Higgins, D.G.; Bleasby, A.J. and Fuchs, R., (1992), "CLUSTAL V : improved software for multiple sequence alignment", CABIOS 8 (2) 189-191

Jones, R., (1992) "Sequence pattern matching on a massively parallel computer", CABIOS (8), 377-383

Karlin, S. and Altschul, S. (1990), "Method for assesing the Statical Significance of Molecular Sequence Features by using General Scoring Schemes", Proc.Natl.Acad.Sci.USA (87) 2264-2268

Lipman D.J. and Pearson W.R. (1985), "Rapid and sensitive protein similarity search", Science, (227), 1435-1444

Lipman, D.J.; Altschul, S.F. and Kececioglu, J.D. (1989), "A tool for multiple sequence alignment", Proc.Natl.Acad.Sci. USA (86) 4412-4415

Livingstone, C.D and Barton, G.J. (1993), "Protein sequence alignments : a strategy for the hierarchical analysis of residue conservation", CABIOS 9 (6), 745-756

Maizel, J.V. and Lenk, R.P, (1981), "Enhanced graphic matrix analysis of nucleic acid and protein sequences", Proc.Natl.Acad.Sci.USA 78(12) 7665-7669

Miller Webb (1993) Building multiple alignments from pairwise alignments. CABIOS (9) 2, 169-176.

Murata,M.; Richardson, J.S.; and Sussman, J.L; (1985), "Simultaneous comparison of three protein sequences", Proc.Natl.Acad.Sci.USA, (82), 3073-3077

Needleman S.B. and Wunsch C.D. (1970) "A general method applicable to the search for similarities in the amino acid sequence of two proteins". J.Mol.Biol 48, 443-453

Pearson W.R. and Lipman D.J.; (1988) "Improved tools for biological sequence comparison", Proc.Natl,Acad.Sci. USA (85), 2444-2448

Pearson W.R. (1990), "Rapid and sensitive sequence comparison with FASTP and FASTA", Methods in Enzymology (183),68-93

Posfai J., Bhagwat,A.S, Posfai G., and Roberts, R.J. (1989) "Predictive motifs derived from cytosine methyltransferases". NAR (17), 2421-2435

Schuler, G.D.; Altschul, S.F. and Lipman, D.J. (1991), "A workbench for multiple alignment construction and analysis", Proteins Struct. Func.Genet. 9(3) 180-190

Sellers, P.H. (1974) "On the theory and computation of evolutionary distances SIAM", J.Appl.Maths, (26), 787-793

Sellers, P.H. (1980) "The theory and computation of evolutionary distances: Pattern recognition", Journal of Algorithms (1), 359-373

Smith, T.F. and Waterman M.S.(1981). "Comparison of Biosequences". Adv.in Aplied Mathematics, (2), 482-489

Smith, R.F. and Smith. T.F. (1992), "Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for use in comparative protein modelling", Protein Eng, 5 (1) 35-41

Taylor, W.R. (1987), "Multiple sequence alignment by a pairwise algorithm"; CABIOS (3) 81-87

Thompson, J.D.; Higgins, D.G.; and Gibson T.J. (1994), "CLUSTAL-W : improving the sensitivity of progressive multiple sequence alignment through sequence wighting position specific gap penalties and weight matrix choice", NAR 22 (22) 4673-4680

Wilbur, W.J. and Lipman, D.J. (1983), "Improved tools for biological sequence comparison", Proc.Natl.Acad.Sci. USA (80), 726-730