FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS
DEPARTAMENTO DE BIOTECNOLOGÍA Y TECNOLOGÍA ALIMENTARIA
UNIVERSIDAD ARGENTINA DE LA EMPRESA

Bioinformática

ANÁLISIS COMPUTACIONAL DE SECUENCIAS

Dr. Lucas L. Maldonado (PhD)

Lic. Biotechnologist and Molecular Biologist

Bioinformatics and genomics specialist

CONICET
Fac. de Medicina - UBA
Fac. de Ciencias Exactas y Naturales – UBA

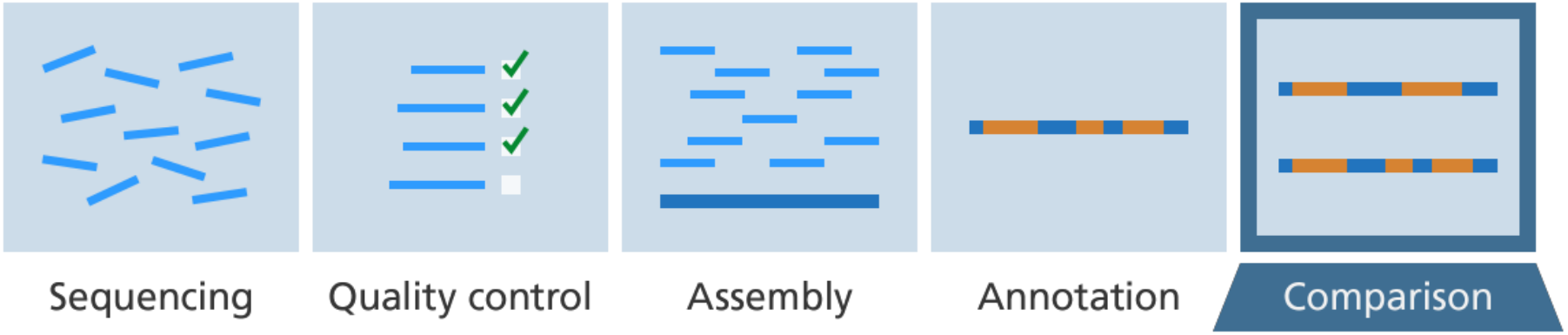
lucamaldonado@uade.edu.ar
lmaldonado@fmed.uba.ar
lucas.l.maldonado@gmail.com.ar

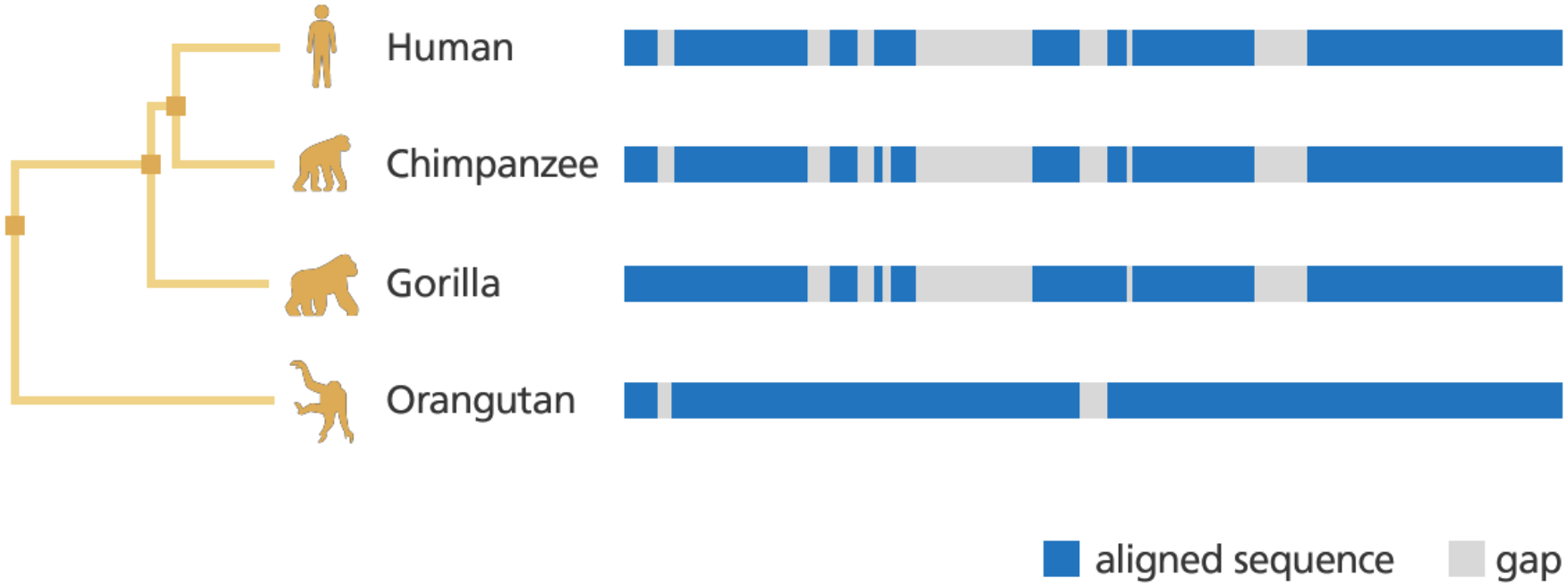
Alineamiento de secuencias

Principios: que es? Para que?

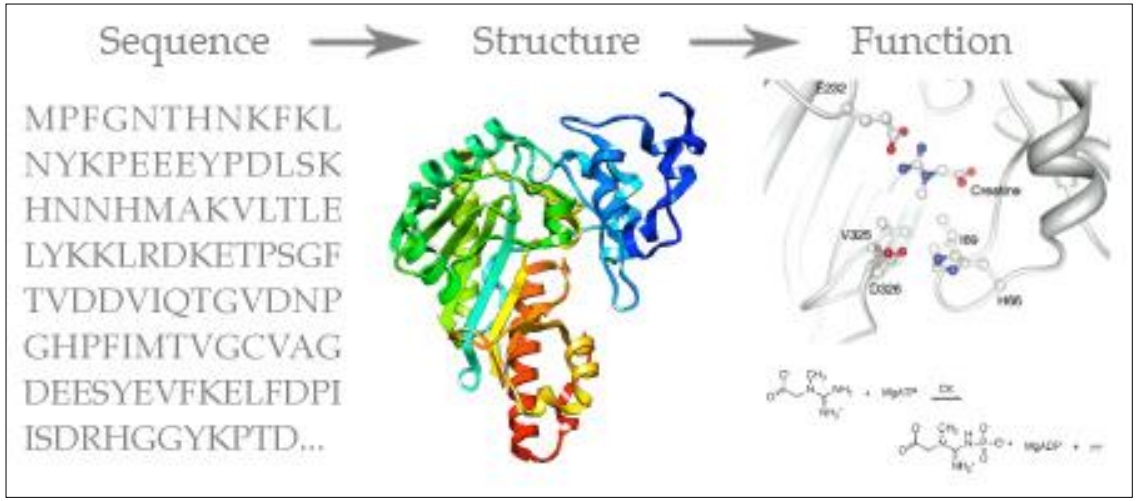
- **Algoritmos**
- **Dot-Plots**
- **NW Puntuación: Simple y Matrices**
- **Programas: BLAST, Fasta**
- **Alineamiento Múltiple**

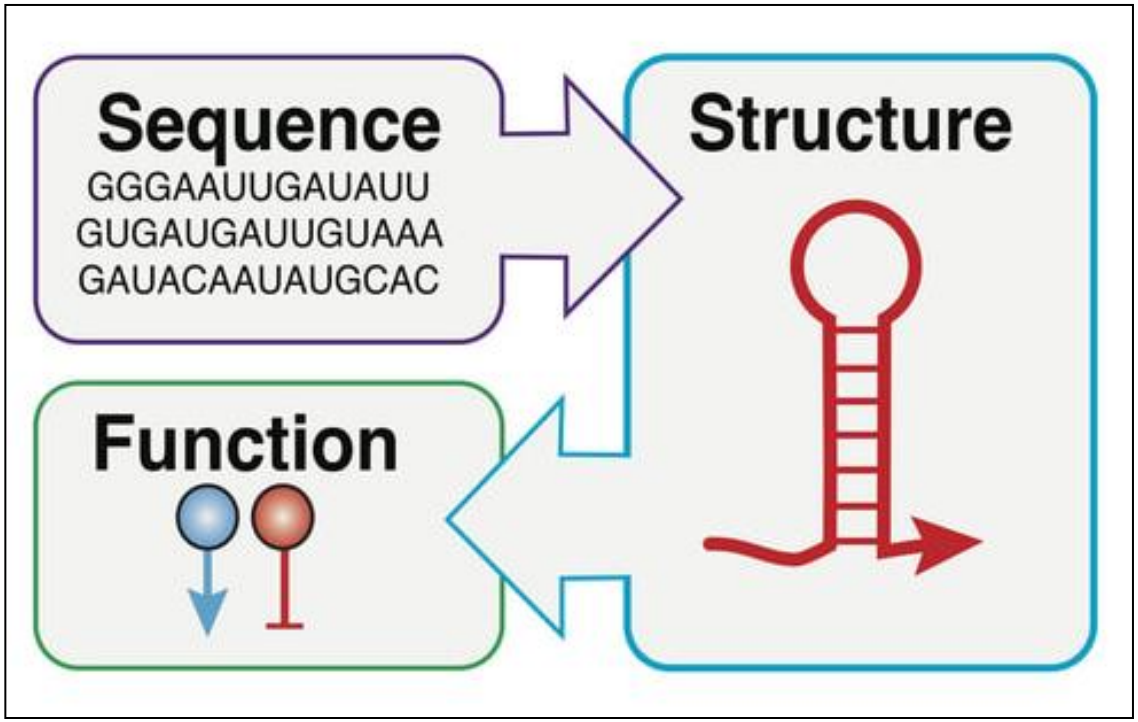
Comparación de secuencias





Comparación de secuencias





Las secuencia del ADN determina la secuencia de una proteína.

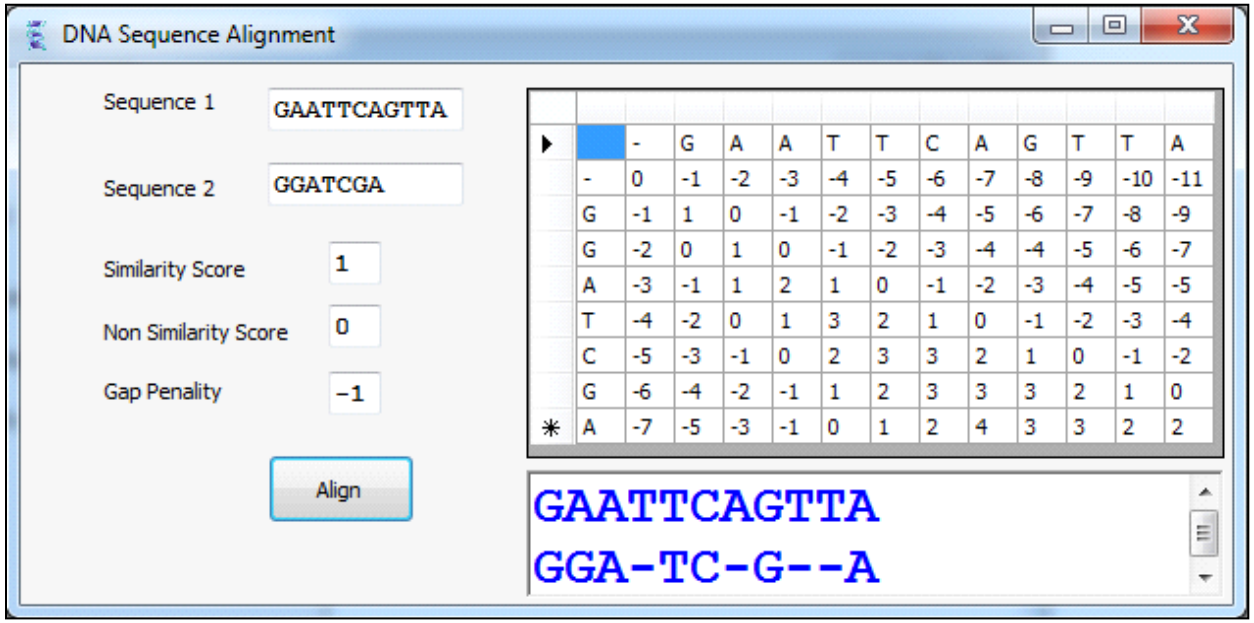
La secuencia de una proteína determina su estructura 3D.

La estructura 3D de una proteína determina su función biológica.

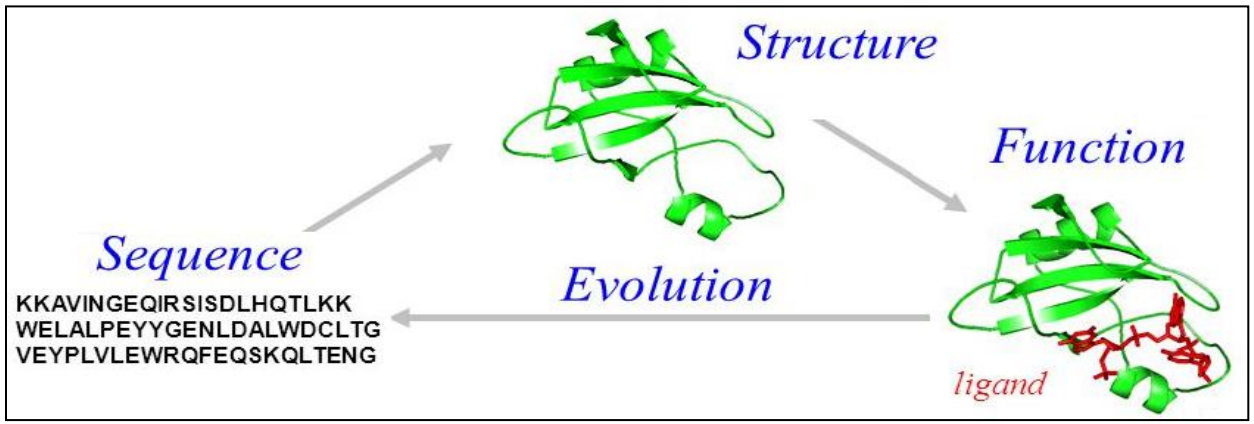
Por tanto, es muy probable que secuencias similares den lugar a proteínas con estructura y función parecidas.

Secuencia → Estructura → Función

Comparación de secuencias



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



- Similar sequence leads to **similar structure**
- Similar structure leads to **similar function**

El <u>alineamiento</u> de secuencias es la técnica que permite <u>establecer el grado de similitud</u> que hay entre ellas.

Cuando el grado de similitud entre dos o más secuencias es elevado, existe una probabilidad muy alta de que se trate de <u>secuencias homólogas</u>.

El alineamiento de secuencias es una herramienta básica de la bioinformática porque <u>permite obtener información funcional, estructural y evolutiva</u>

Comparación de secuencias

1.- Alineamiento global

Un alineamiento global se extiende por toda la longitud de la secuencia

```
--T--CC-C-AGT--TATGT-CAGGGGACACG--A-GCATGCAGA-GAC
  |   ||  |   ||   | |  |  |||     ||  |   | |   | ||||    |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG--T-CAGAT--C
```

tiomología

2.- Alineamiento local

Un alineamiento local se limita a una región concreta de la secuencia

```
gagcatgcagagactccCAGTTATGTCAGgggacacgagcatgca
                 ||||||||||||
gccgccgtcgttttcagCAGTTATGTCAGatcgccgccgtcgttt
```

Motivos conservados

3.- Alineamiento semiglobal

Un alineamiento semiglobal se produce entre el final de una secuencia y el inicio de otra

```
                 CAGTTATGTCAGgggacacgagcatgca
                 ||||||||||||
gccgccgtcgttttcagCAGTTATGTCAG
```

Ensamblaje de secuencias

Tipos de alineamiento de dos secuencias

Métodos de alineamientos y algoritmos

Existen diversos métodos para el alineamiento de dos secuencias:

1.- El algoritmo de la fuerza bruta

2.- Matrices de puntos (*dot plots*)

3.- El algoritmo de programación dinámica

4.- Métodos heurísticos (FASTA, BLAST)

Usan “palabras” para anclar dos secuencias

Estrategias para alinear dos secuencias

Métodos de alineamientos

A mano:

se mueven las secuencias a mano y se observan... (no muy practico!)

Dot plot (Método gráfico)

se gráfica en una matriz de a “ventanas”

Dynamic programming

Smith-Waterman
Needle-Wunsch

} (lento, óptimo, garantiza el mejor posible alineamiento)

Heuristic methods

(rápido, se aproxima a un buen resultado. No lo garantiza)

BLAST and FASTA: Usan “palabras” para anclar dos secuencias!

Métodos de alineamientos y algoritmos

En resumen...

Una vez fijado un sistema de puntuación

- Matriz de substitución (Identidad, PAMxx, BLOSUM…)
- Coste de la apertura y de la extensión de “gaps”

- El problema se reduce a encontrar el alineamiento optimo (o el mejor, no?)
- Se define el alineamiento óptimo entre dos secuencias como aquel cuya puntuación es máxima entre todos los posibles alineamientos.

¿Como encuentro el alineamiento de mayor puntuación?

Métodos de alineamientos y algoritmos

Resumamos…..

Hasta acá vimos:

Muchos conceptos:
Alineamientos Local y global, alineamientos de a pares
Conceptos de homología y similitud y diferencias entre alinear nucleótidos y aminoácidos, etc
Algoritmos de alineamiento: Fuerza bruta, dot-plot
Estrategias de Scoring para la elección del mejor alineamiento
Matrices de puntaje de ADN y proteínas y gaps

¿Que nos falta?:

Más algoritmos de alineamiento:

- **Programación dinámica:**
  - Smith-Waterman: alineamientos locales
  - Needleman-Wunsch: alineamientos globales
- **Heuristica**: BLAST y FASTA

Alineamientos multiples

Métodos de alineamientos y algoritmos

Programación dinámica

Alineamiento óptimo

The optimal alignment of two sequences is one that finds the longest segment of high sequence similarity.

Needleman & Wunsch

Alineamiento Global

Smith & Waterman (SW)

Alineamiento local

Dynamic programming algorithms

- Each character along both sequences is evaluated. At each position.
- There are four possibilites: each one is scored according to a matrix

  - Identity (Match)
  - Substitution (Mismatch)
  - deletion in sequence 1 (Gap)
  - deletion in sequence 2 (Gap)

Dynamic programming

The quality of the alignment depends on the final score

- **Quality =** matches - (mismatches + gap penalty)
- The program will find the alignment with the **highest quality**.
- The choice between gaps and substitutions is made to give the higher quality of the two.

Dynamic programming

- Hace falta encontrar una fórmula recurrente (algoritmo) que vaya encontrando la solución de los subproblemas

- Se empieza resolviendo el subproblema más trivial. La solución de cada subproblema se va guardando en la memoria del ordenador. Al final, se obtiene la solución del problema global

- El algoritmo trata de maximizar la puntuación final del alineamiento haciendo coincidir el mayor número posible de emparejamientos de valor elevado y minimizando el número de indels y de emparejamientos de poco valor

Dynamic programming

Se necesitan 2 secuencias y un sistema de puntuación

Garantiza el alineamiento óptimo

Se lleva a cabo en tres etapas:

- **Inicialización**
- **Rellenado de la matriz**
- **Retroceso**

Es costoso en términos de tiempo de computación y de memoria de ordenador

Dynamic prgramming

- Tenemos un método (algoritmo) que nos garantiza un alineamiento optimo entre dos secuencias
- Tenemos un sistema de scoring complejo que refleja mejor nuestras ideas biológicas sobre lo que sería un buen alineamiento

Entendamos el algoritmo con el siguiente ejemplo:

Dynamic programming

Un ejemplo:

Construir un alineamiento óptimo entre estas dos secuencias

G A T A C T A

G A T T A C C A

Utilizando las siguientes reglas de scoring:

Match: +1

Mismatch: -1

Gap: -1

$$D_{i,j} = \max \begin{cases} D_{i-1,j-1} + s(a_i, b_j) \\ D_{i-1,j} + s(a_i, -) \\ D_{i,j-1} + s(-, b_j) \end{cases} = \max \begin{cases} D_{i-1,j-1} + 1 & a_i = b_j \\ D_{i-1,j-1} + -1 & a_i \neq b_j \\ D_{i-1,j} + -1 & b_j = - \\ D_{i,j-1} + -1 & a_i = - \end{cases}$$

Construir un alineamiento global óptimo entre estas dos secuencias

Dynamic programming

Ordenar las dos secuencias en una matriz bidimensional

Los vértices de cada celda se encuentran entre letras (bases).

Needleman & Wunsch (1970)

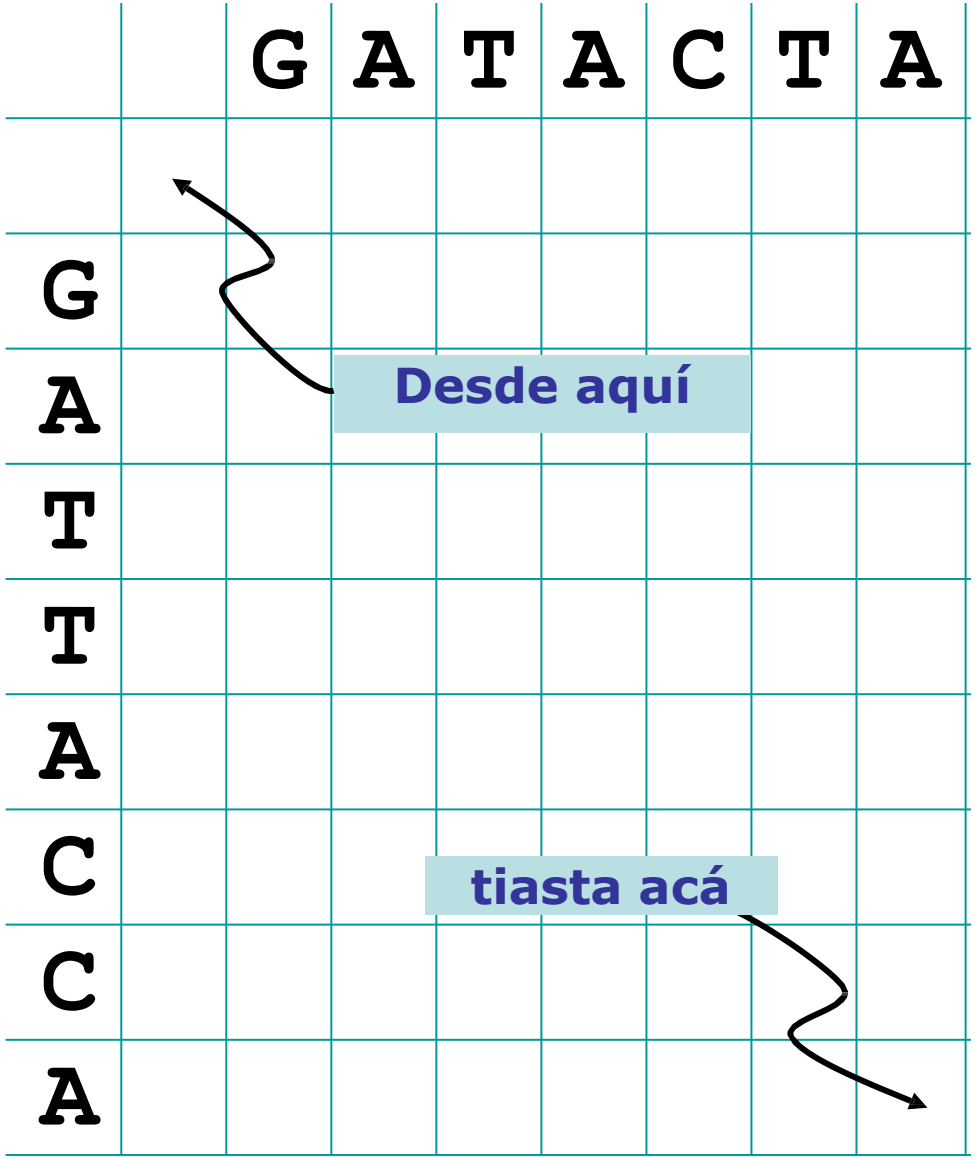
		G	A	T	A	C	T	A
G								
A								
T								
T								
A								
C								
C								
A								

Dynamic programming

El objetivo es encontrar la ruta (path) óptimo

Alineamiento global

Needleman & Wunsch (1970)



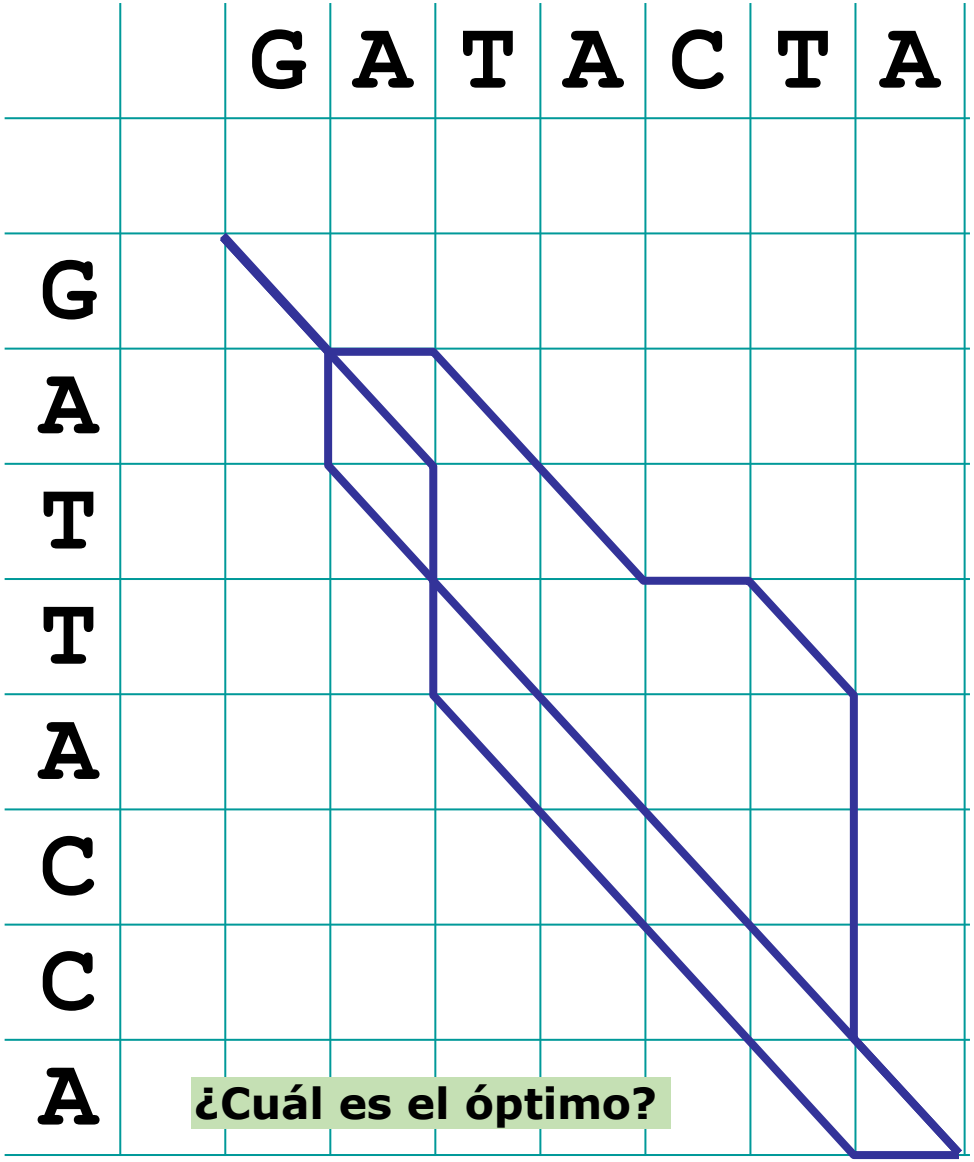
Slides Dynamic Programming: tiugues Sicotte (NCBI)

Dynamic programming

Cada path corresponde a un alineamiento único

Alineamiento global

Needleman & Wunsch (1970)



Slides Dynamic Programming: tiugues Sicotte (NCBI)

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)

		G	A	T	A	C	T	A
	0	-1						
G	-1	+1						
A								
T								
T								
A								
C								
C								
A								

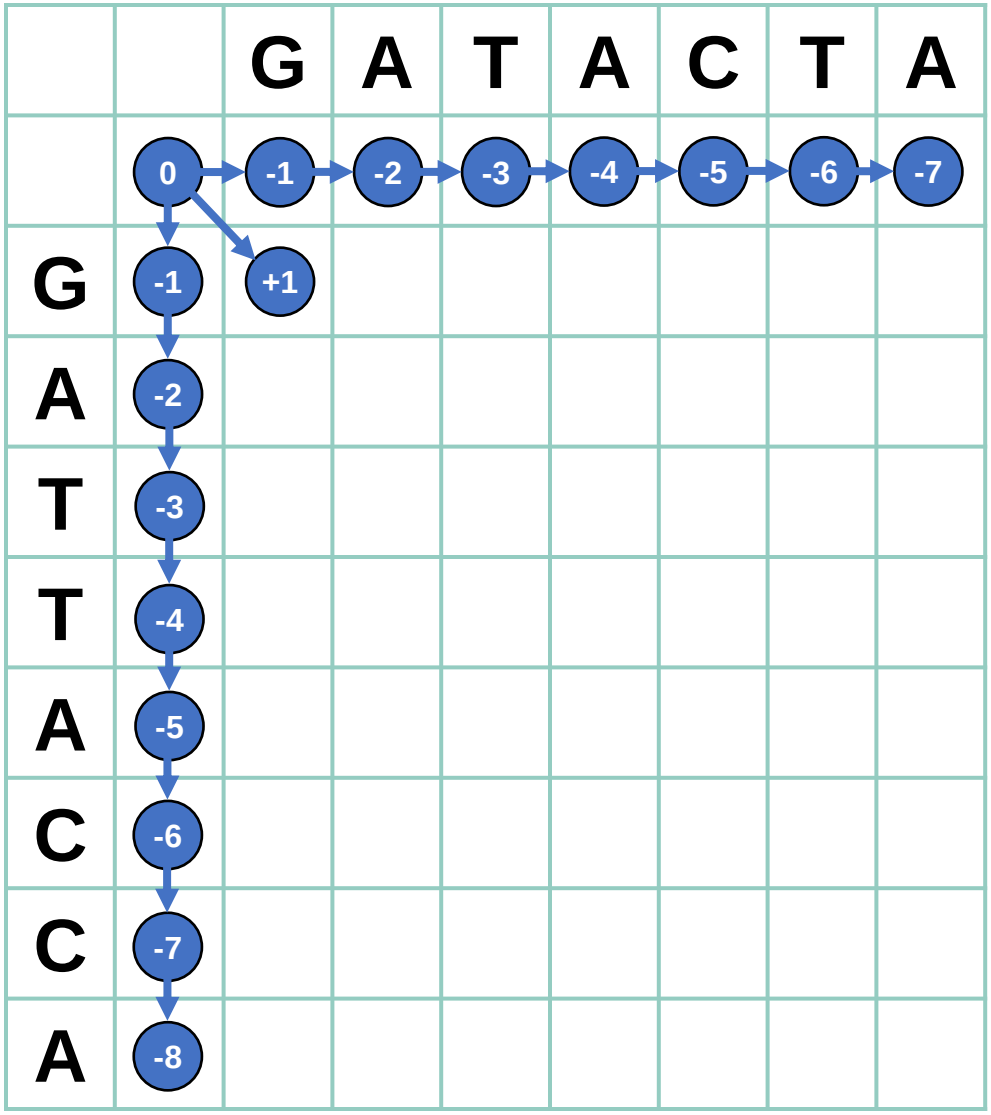
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



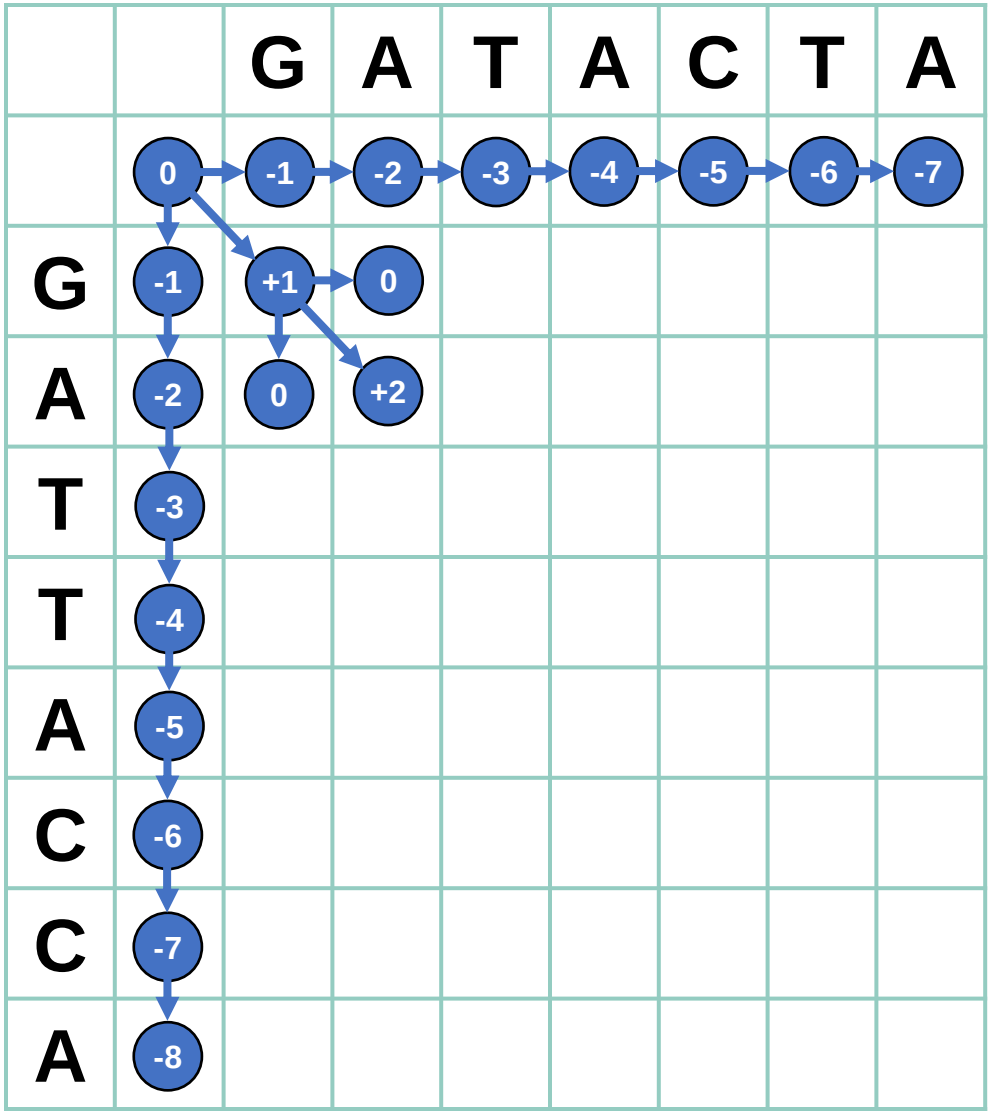
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



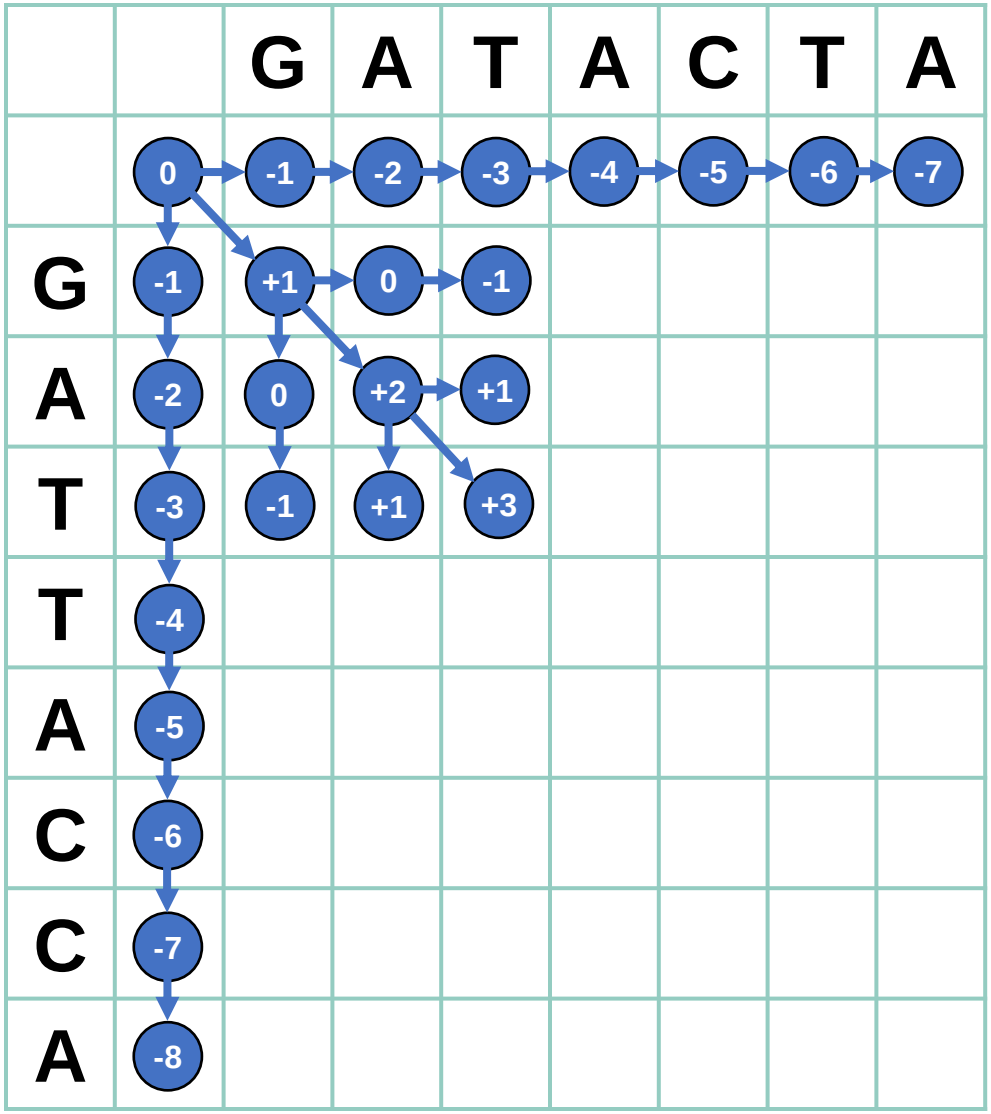
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



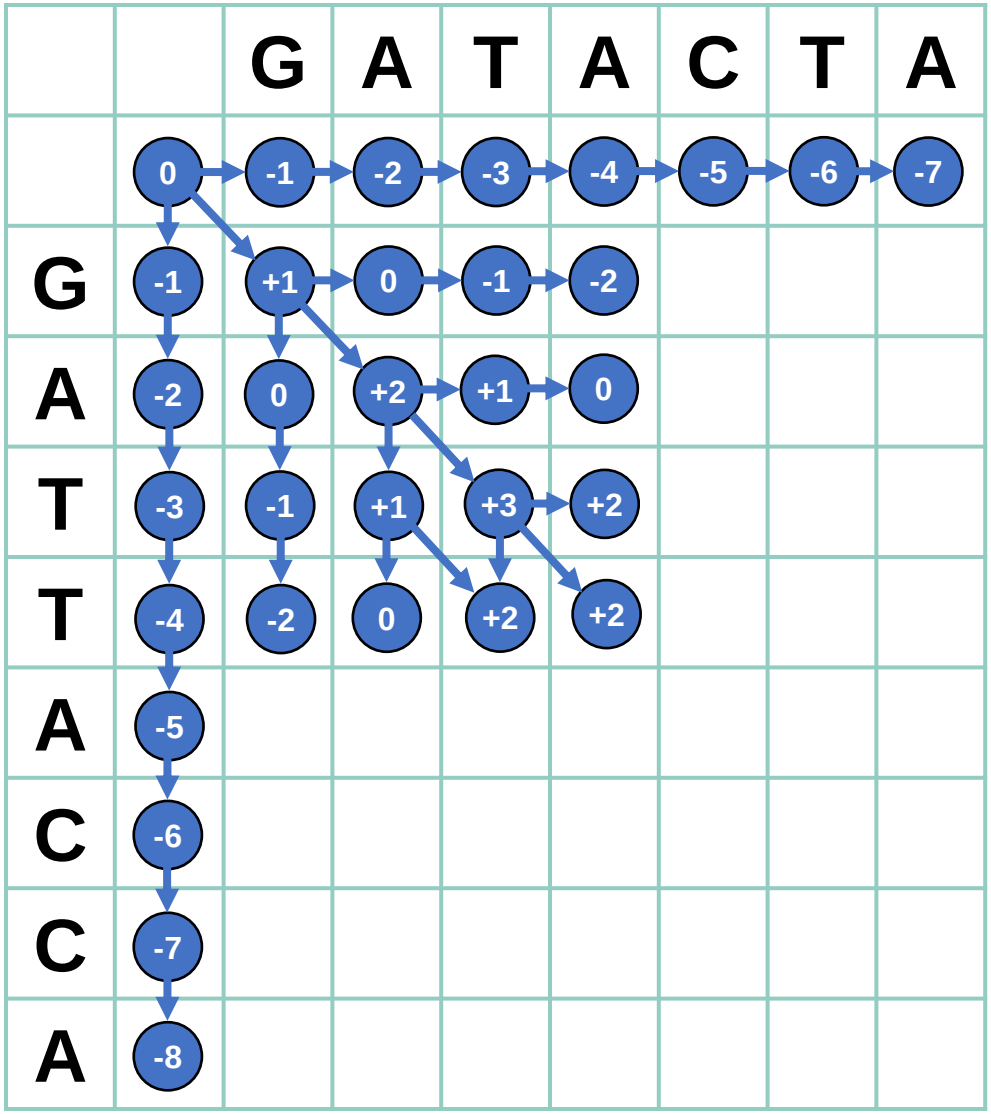
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



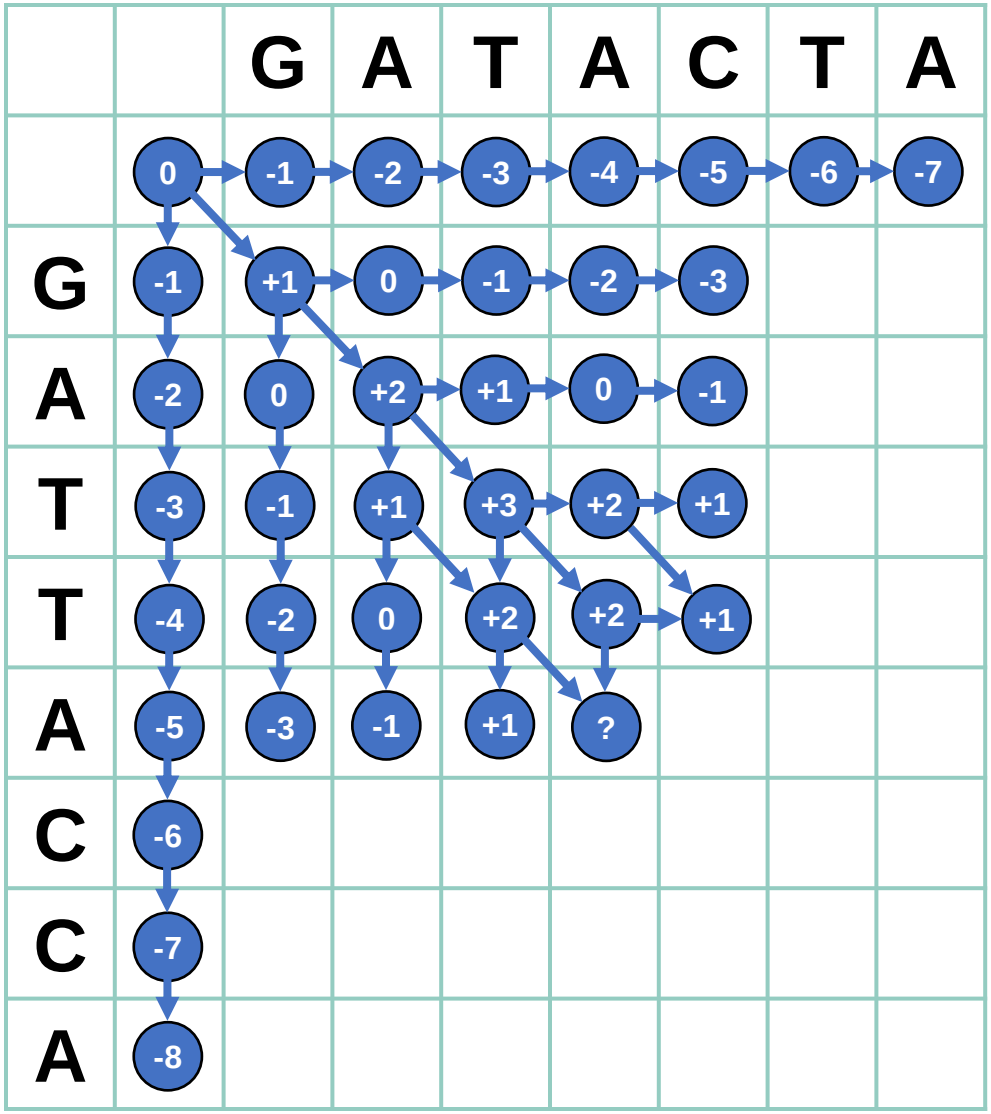
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



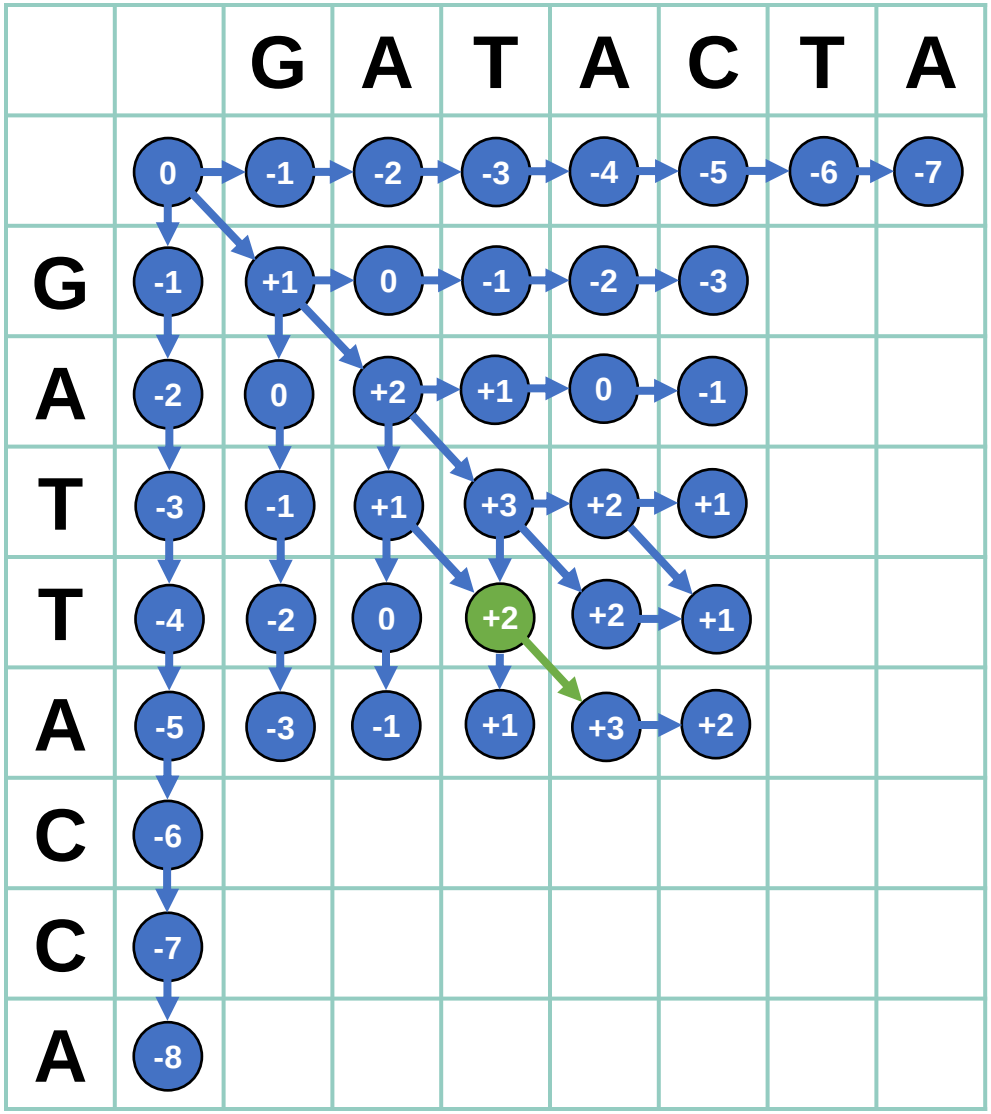
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



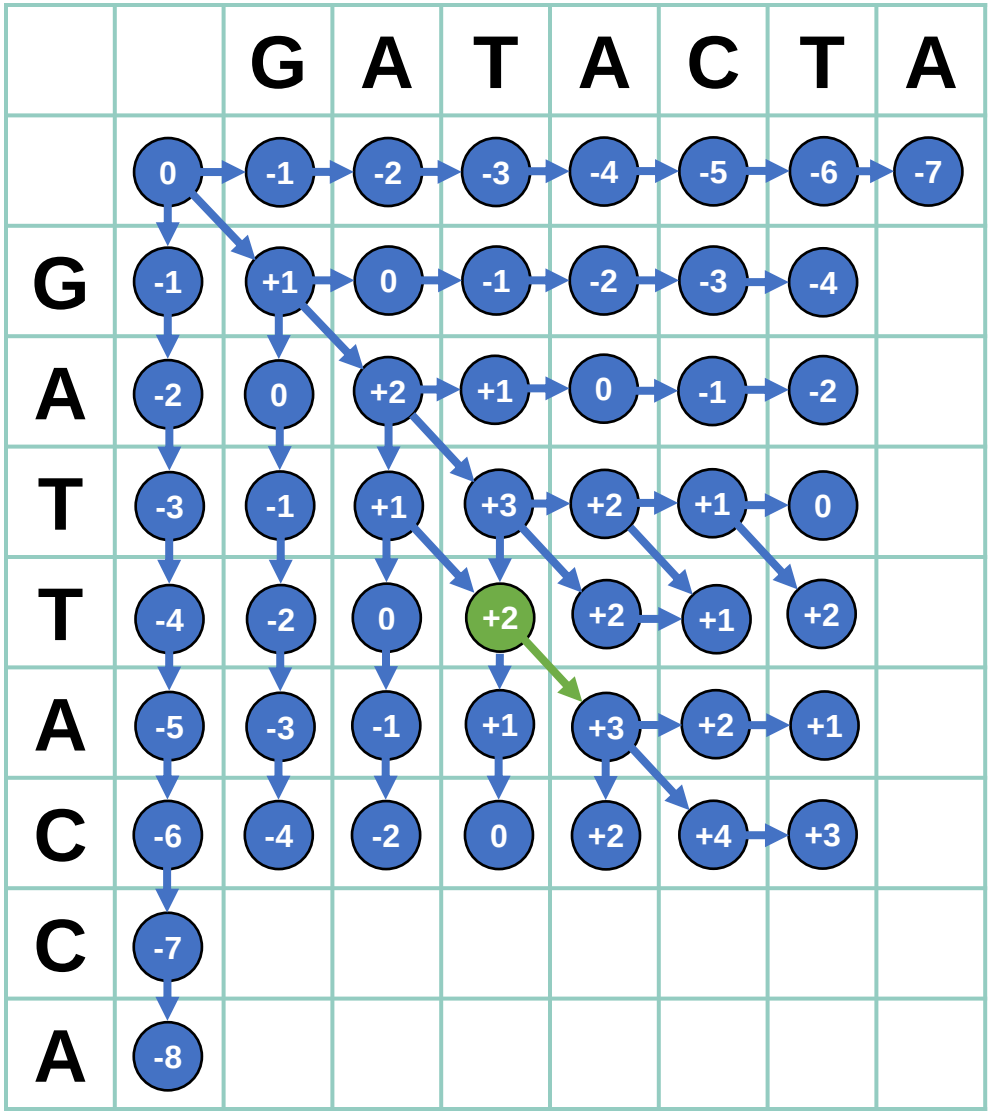
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



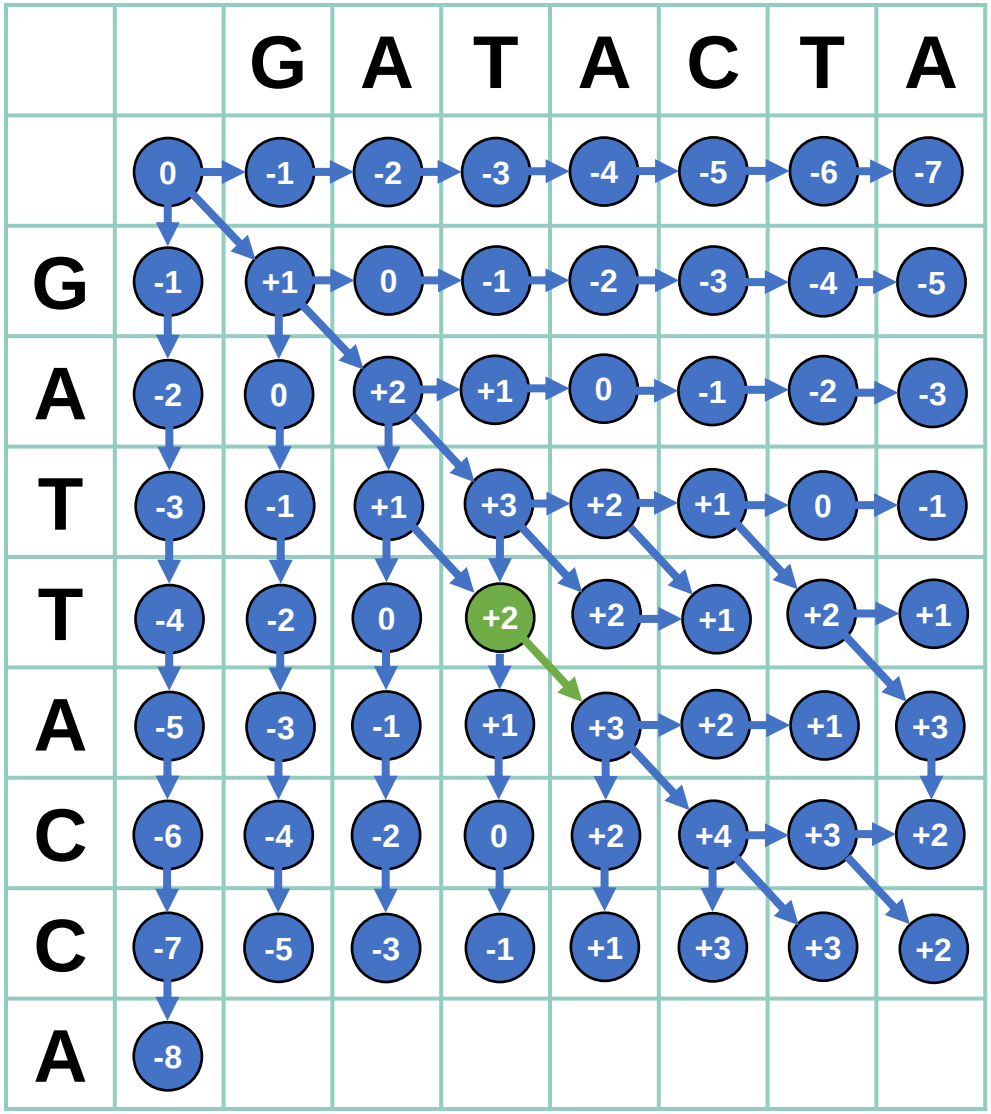
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



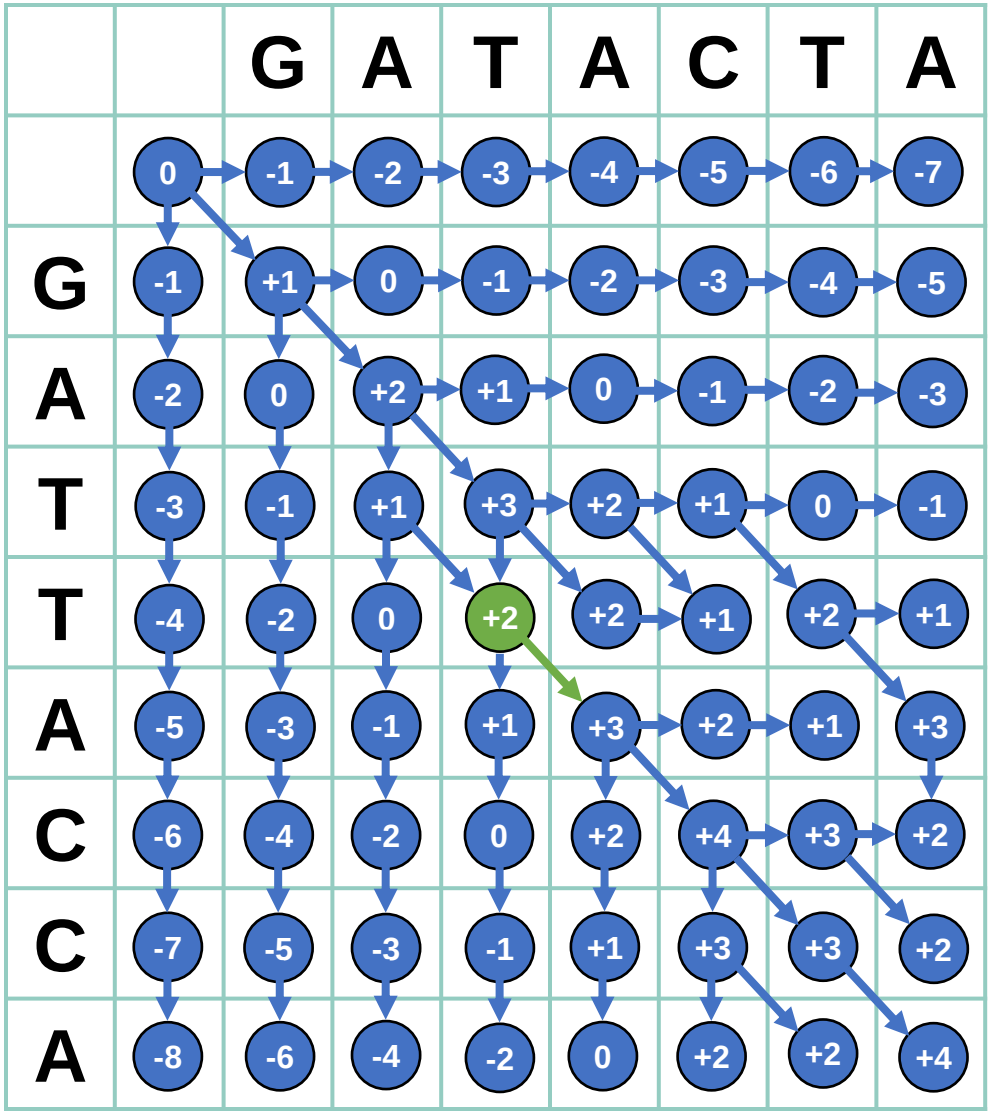
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



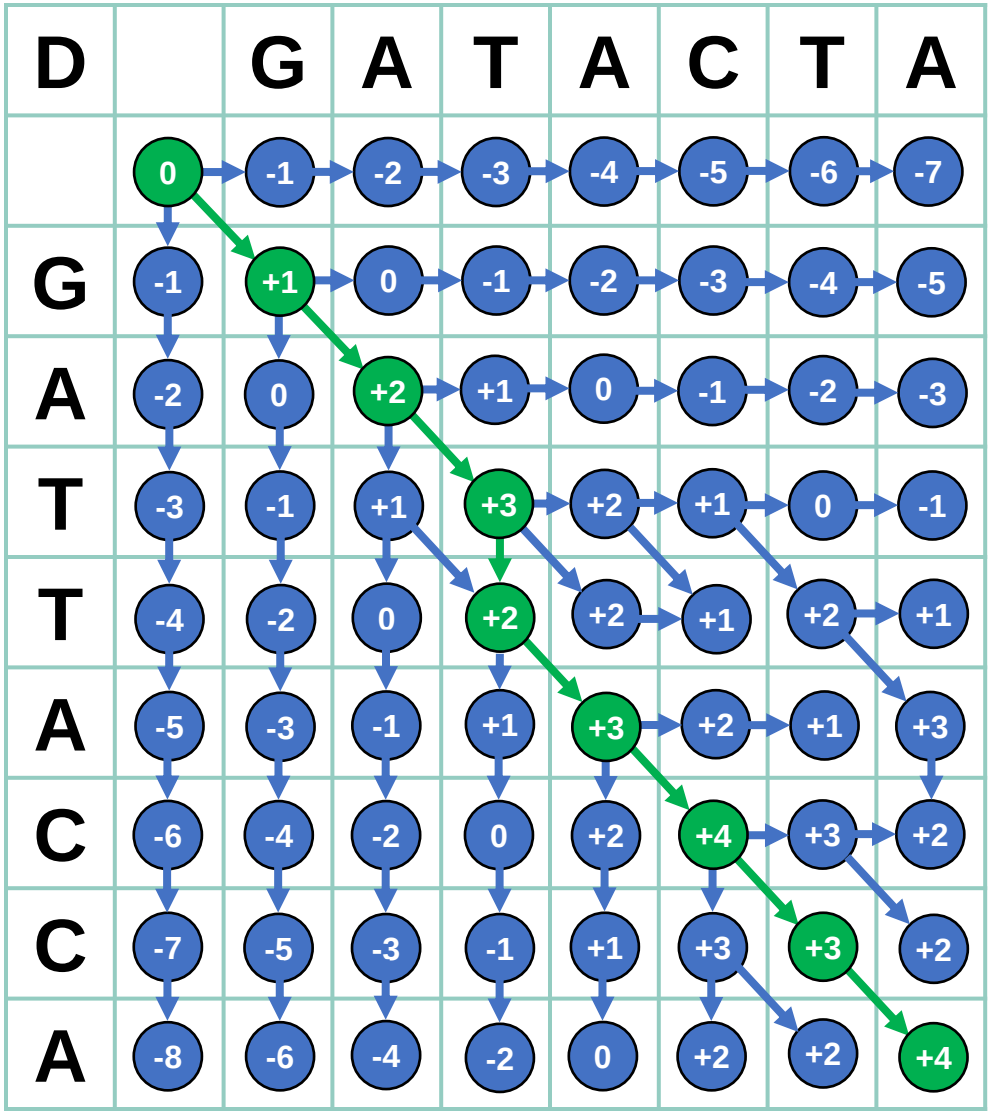
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



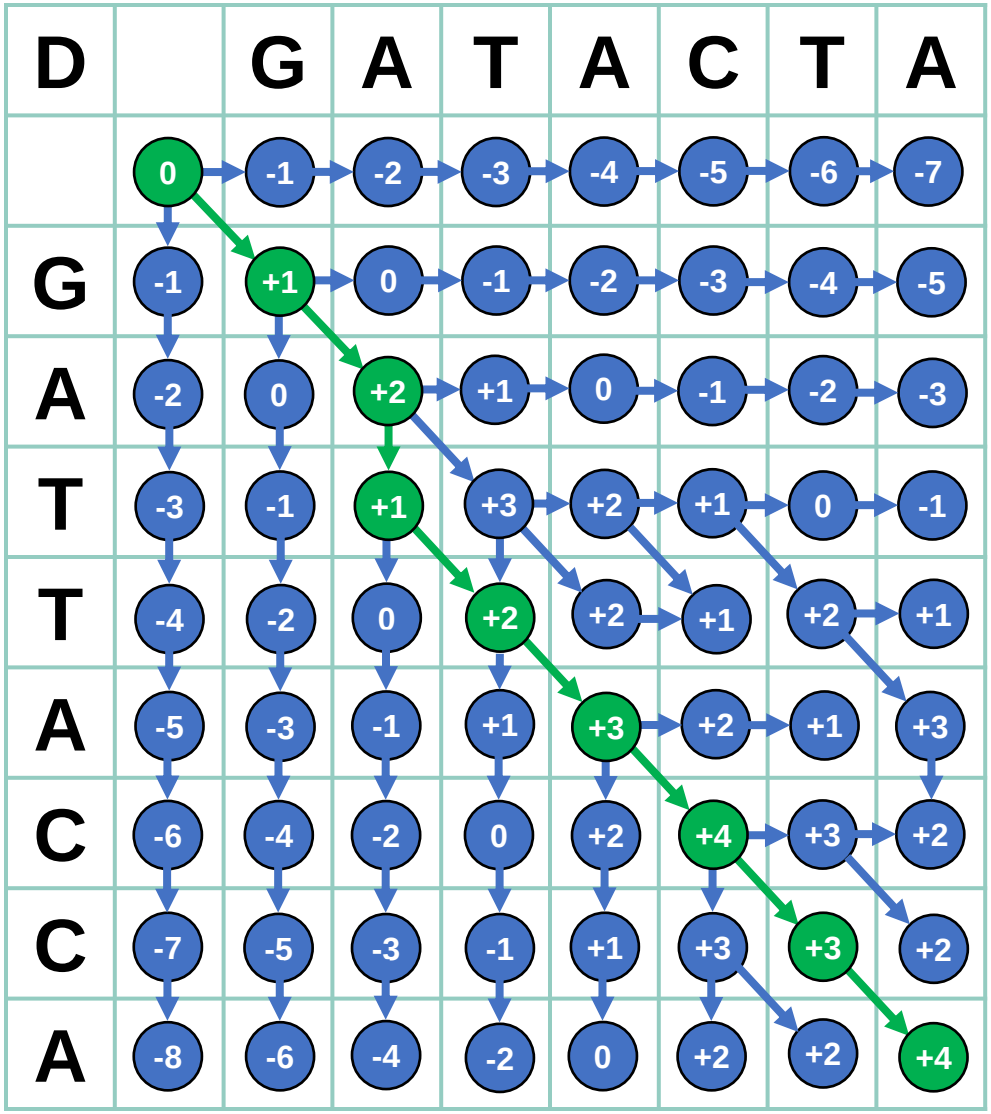
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



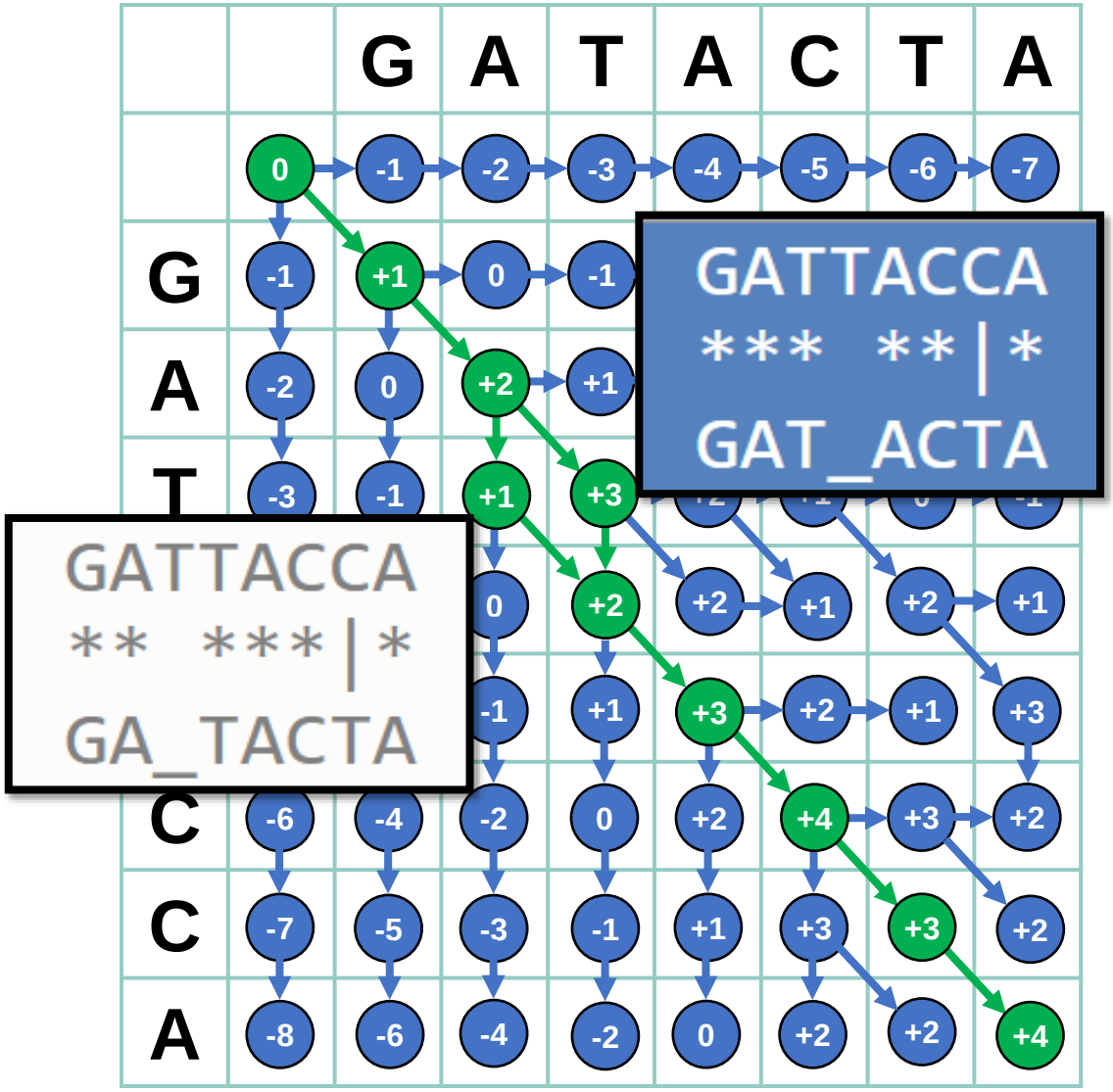
El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Alineamiento Global
Needleman & Wunsch (1970)



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

En resumen

- The alignment is over the entire length of two sequences: the traceback starts from tne lower right corner of the traceback matrix, and completes in the upper left cell of the matrix.

- The Needleman-Wunsch algorithm works in the same way regardless of the length or complexity of sequences, and *guarantees* to find tne 5est alignment.

- The Needleman-Wunsch algorithm is appropriate for finding the best alignment of two sequences which are *(i)* of the similar length; (*ii)* similar across their entire lengths.

Dynamic programming

Veamos ahora como se usa la programación dinámica para hacer un alineamiento local entre dos secuencias

Smith & Waterman (SW)

Alineamiento local

Algoritmo de Smith & Waterman (SW)

Alineamiento Local

J. Mol. Biol. (1981), **147**, 195–197

Identification of Common Molecular Subsequences

T. F. SMITH and M. S. WATERMAN

The identification of maximally homologous subsequences among sets of long sequences is an important problem in molecular sequence analysis. The problem is straightforward only if one restricts consideration to contiguous subsequences (segments) containing no internal deletions or insertions. The more general problem has its solution in an extension of sequence metrics (Sellers 1974; Waterman *et al.*, 1976) developed to measure the minimum number of "events" required to convert one sequence into another.

These developments in the modern sequence analysis began with the heuristic homology algorithm of Needleman & Wunsch (1970) which first introduced an iterative matrix method of calculation. Numerous other heuristic algorithms have been suggested including those of Fitch (1966) and Dayhoff (1969). More mathematically rigorous algorithms were suggested by Sankoff (1972), Reichert *et al.* (1973) and Beyer *et al.* (1979), but these were generally not biologically satisfying or interpretable. Success came with Sellers (1974) development of a true metric measure of the distance between sequences. This metric was later generalized by Waterman *et al.* (1976) to include deletions/insertions of arbitrary length. This metric represents the minimum number of "mutational events" required to convert one sequence into another. It is of interest to note that Smith *et al.* (1980) have recently shown that under some conditions the generalized Sellers metric is equivalent to the original homology algorithm of Needleman & Wunsch (1970).

In this letter we extend the above ideas to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology). The similarity measure used here allows for arbitrary length deletions and insertions.

Artículo de Smith y Waterman (1981)

Algoritmo de Smith & Waterman (SW)

Alineamiento Local

Introduce tres modificaciones en relación al algoritomo de Needleman-Wunsch:

1.- La penalización por introducir un hueco al comienzo del alineamiento es 0

2.- Cuando un valor de la matriz de puntuación se hace negativo, se pone un 0. Es como si se comenzara un nuevo alineamiento.

3.- La máxima puntuación puede estar en cualquier lugar de la tabla. El retroceso comienza en la casilla con la puntuación más elevada y prosigue hasta alcanzar un 0.

Dynamic programming

Un ejemplo:

Construir un alineamiento óptimo entre estas dos secuencias

G A T A C T A

G A T T A C C A

Utilizando las siguientes reglas de scoring:

Match: +1

Mismatch: -1

Gap: -1

$$D_{i,j} = \max \begin{cases} D_{i-1,j-1} + s(a_i, b_j) \\ D_{i-1,j} + s(a_i, -) \\ D_{i,j-1} + s(-, b_j) \end{cases} = \max \begin{cases} D_{i-1,j-1} + 1 & a_i = b_j \\ D_{i-1,j-1} + -1 & a_i \neq b_j \\ D_{i-1,j} + -1 & b_j = - \\ D_{i,j-1} + -1 & a_i = - \end{cases}$$

Construir un alineamiento Local óptimo entre estas dos secuencias

Características del algoritmo

- Se necesitan 2 secuencias y un sistema de puntuación

- Garantiza el alineamiento óptimo

- Se lleva a cabo en tres etapas:

  - Inicialización
  - Rellenado de la matriz
  - Retroceso

- Es costoso en términos de tiempo de computación y de memoria de ordenador

Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local

		G	A	T	A	C	T	A
	0	0						
G	0	+1						
A								
T								
T								
A								
C								
C								
A								

El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

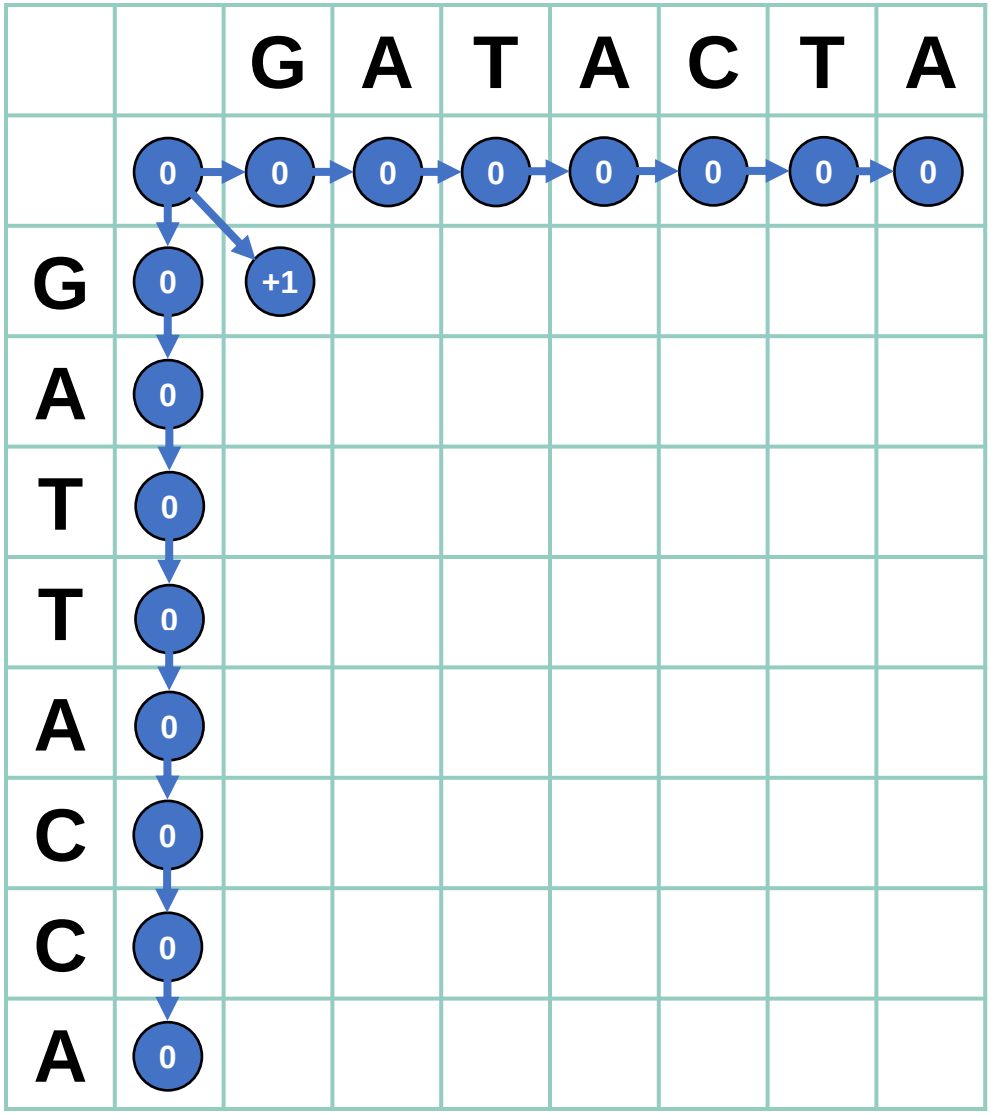
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

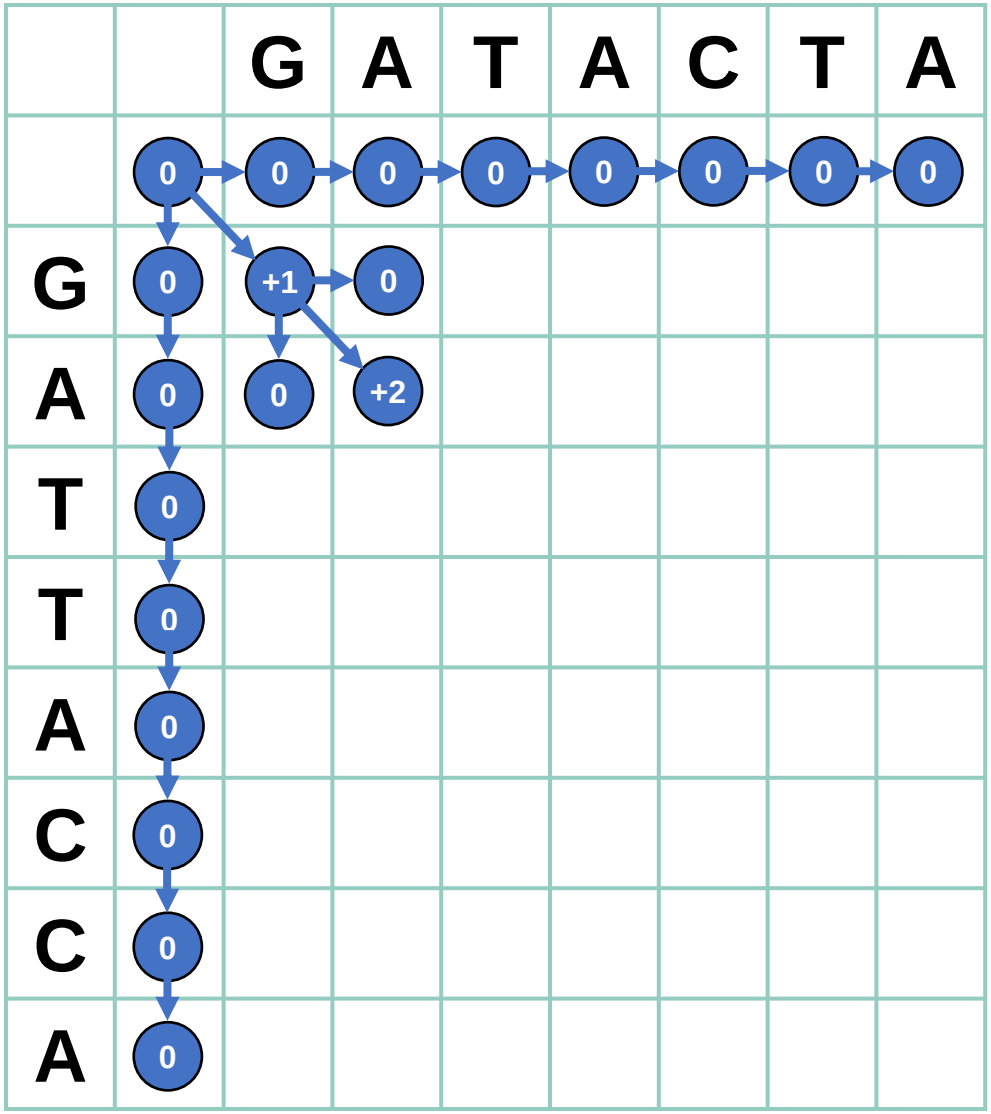
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

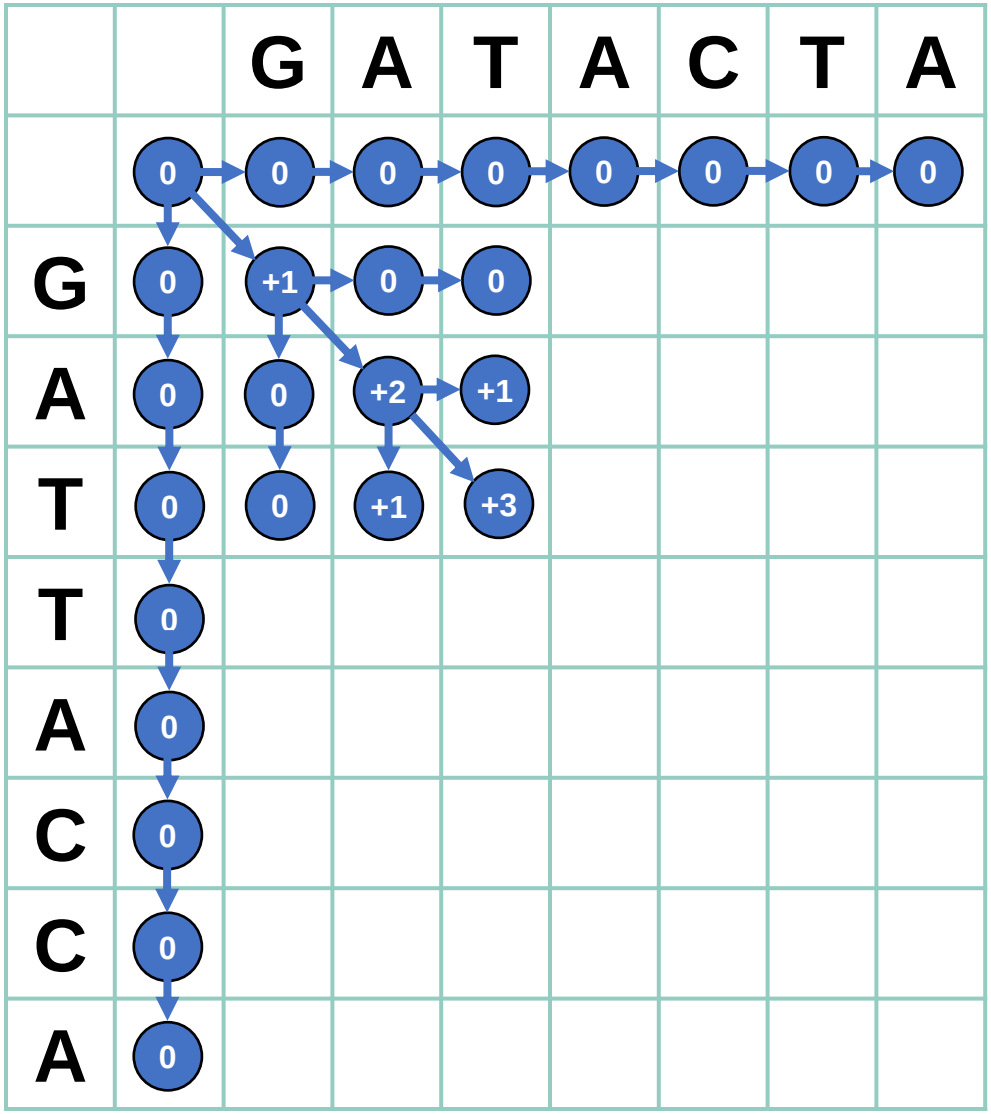
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

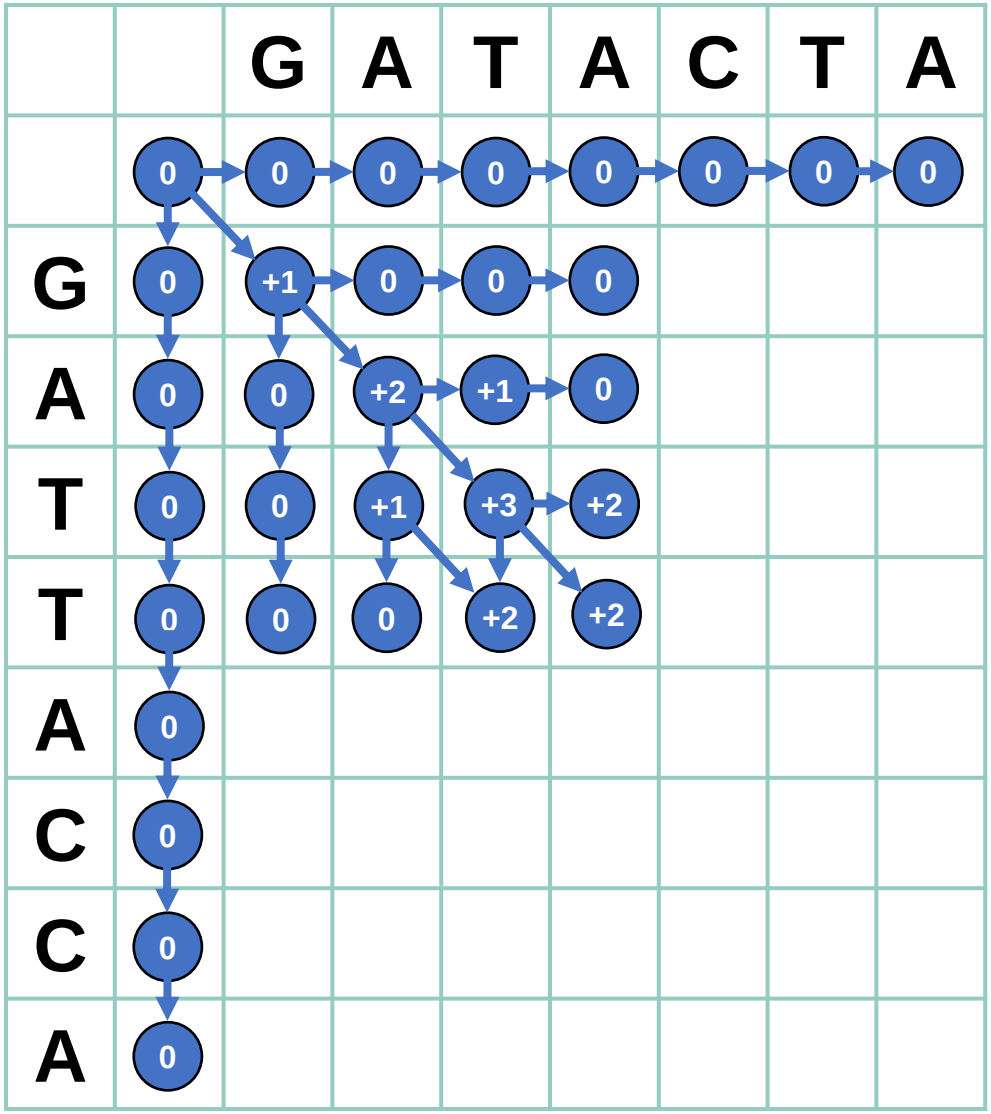
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

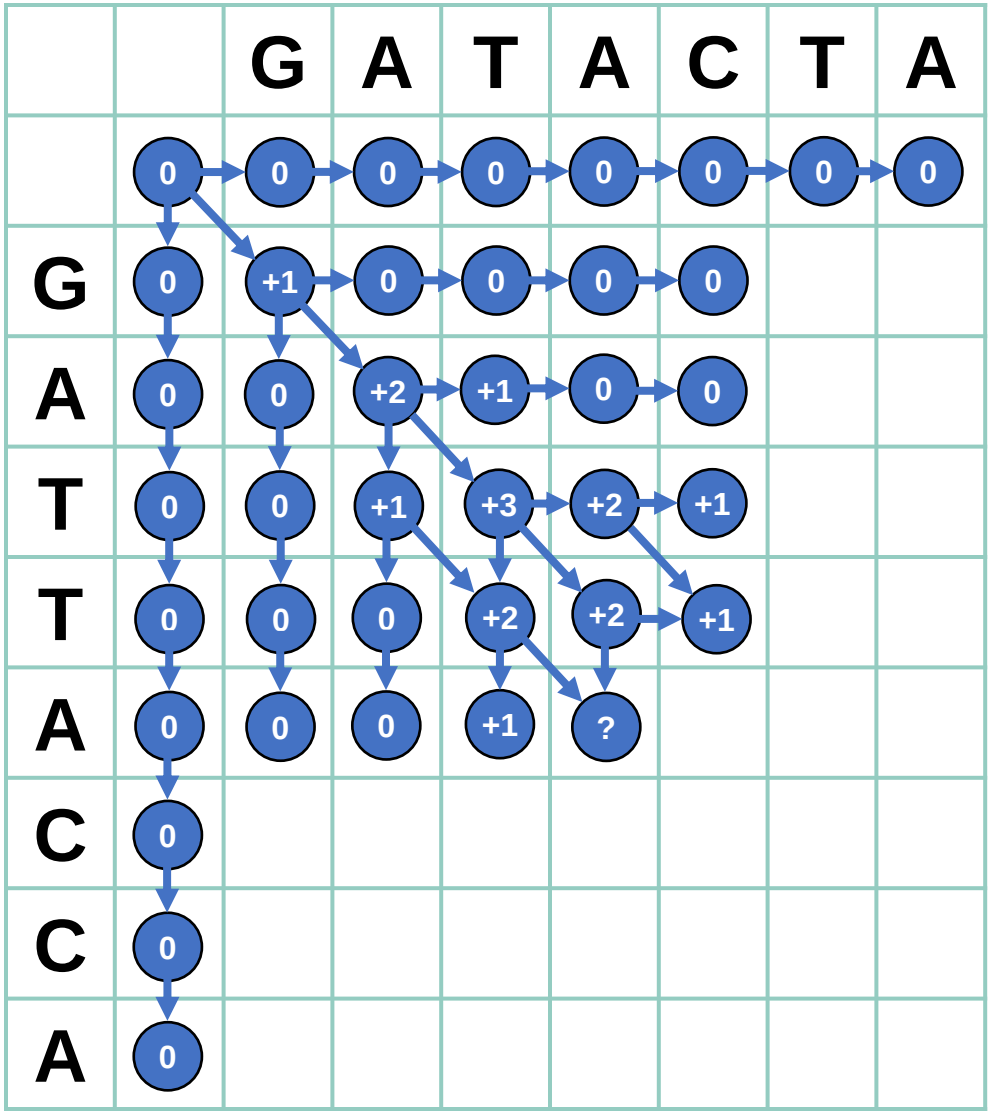
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

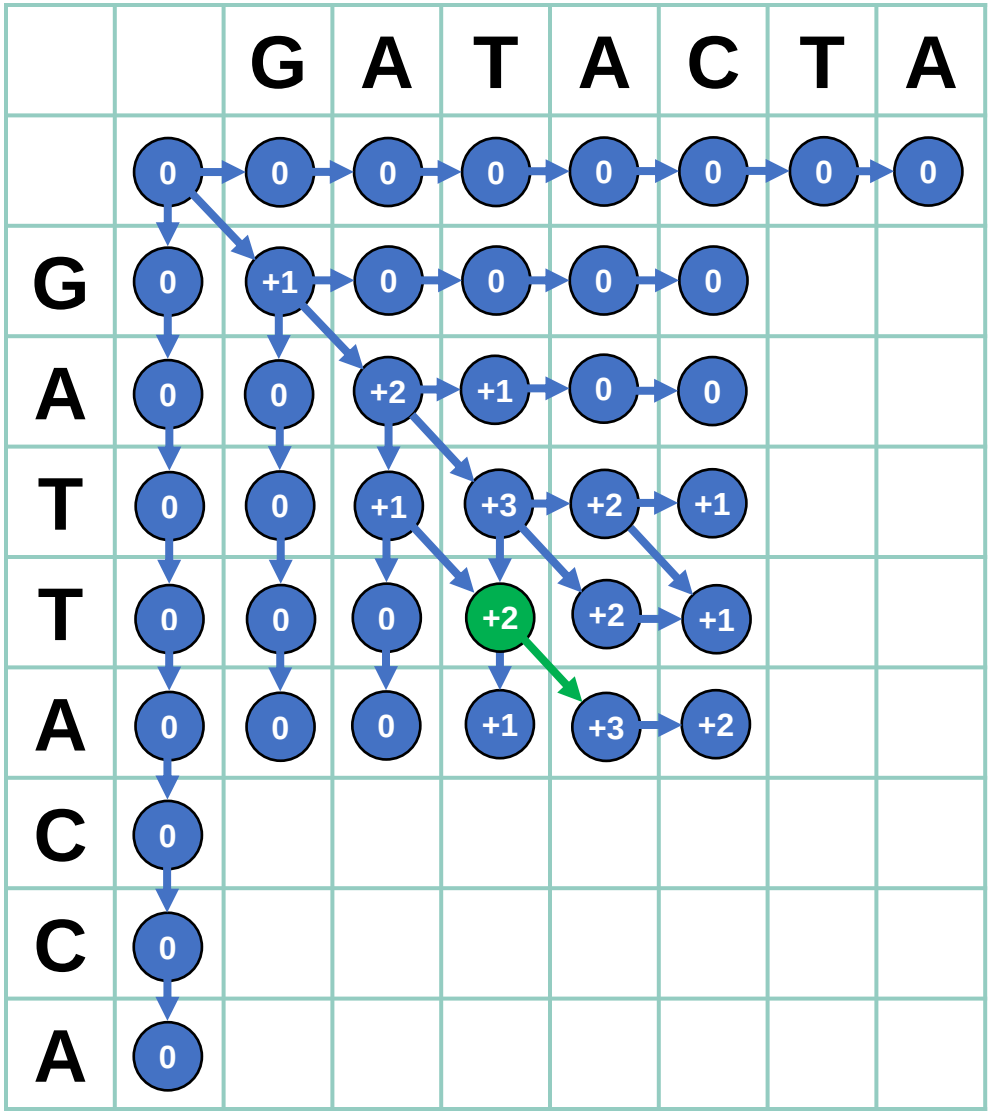
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

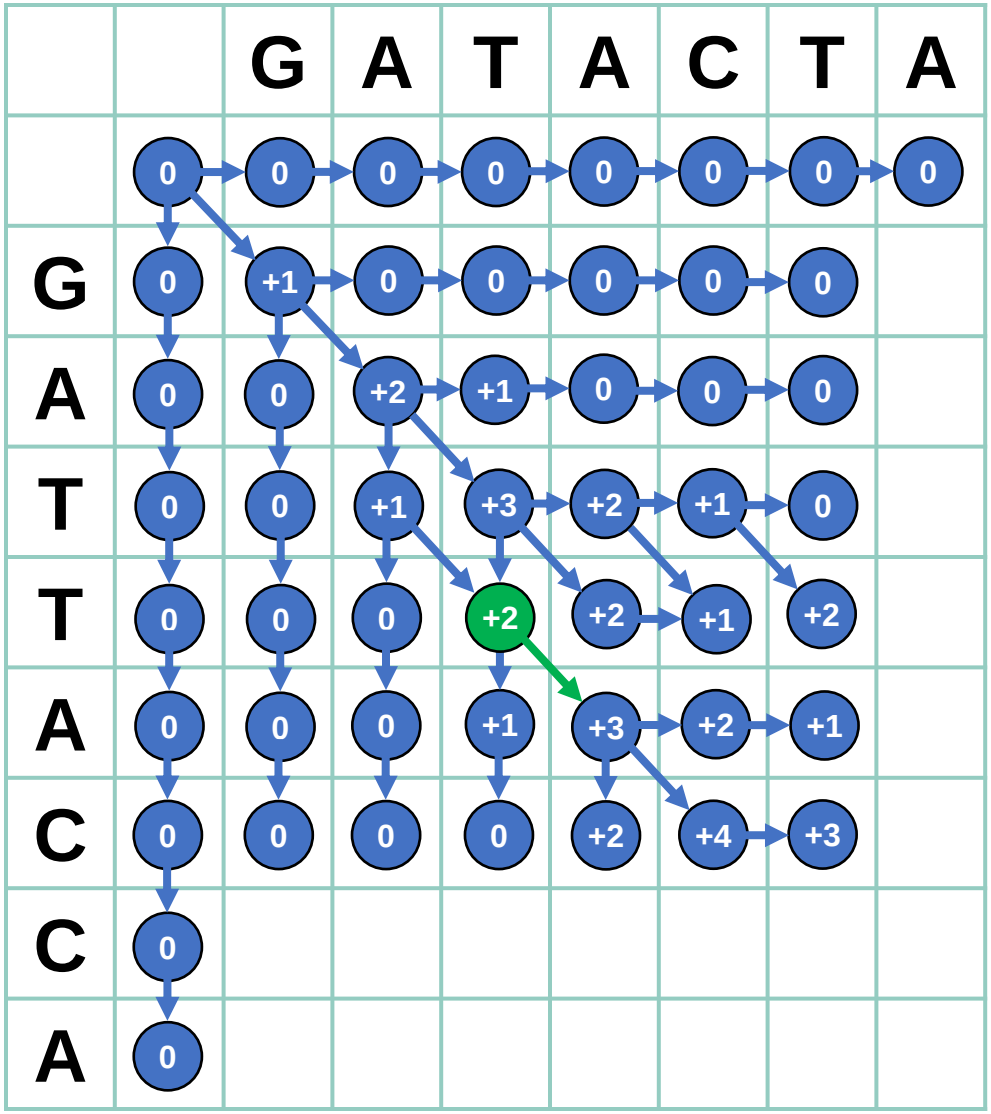
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

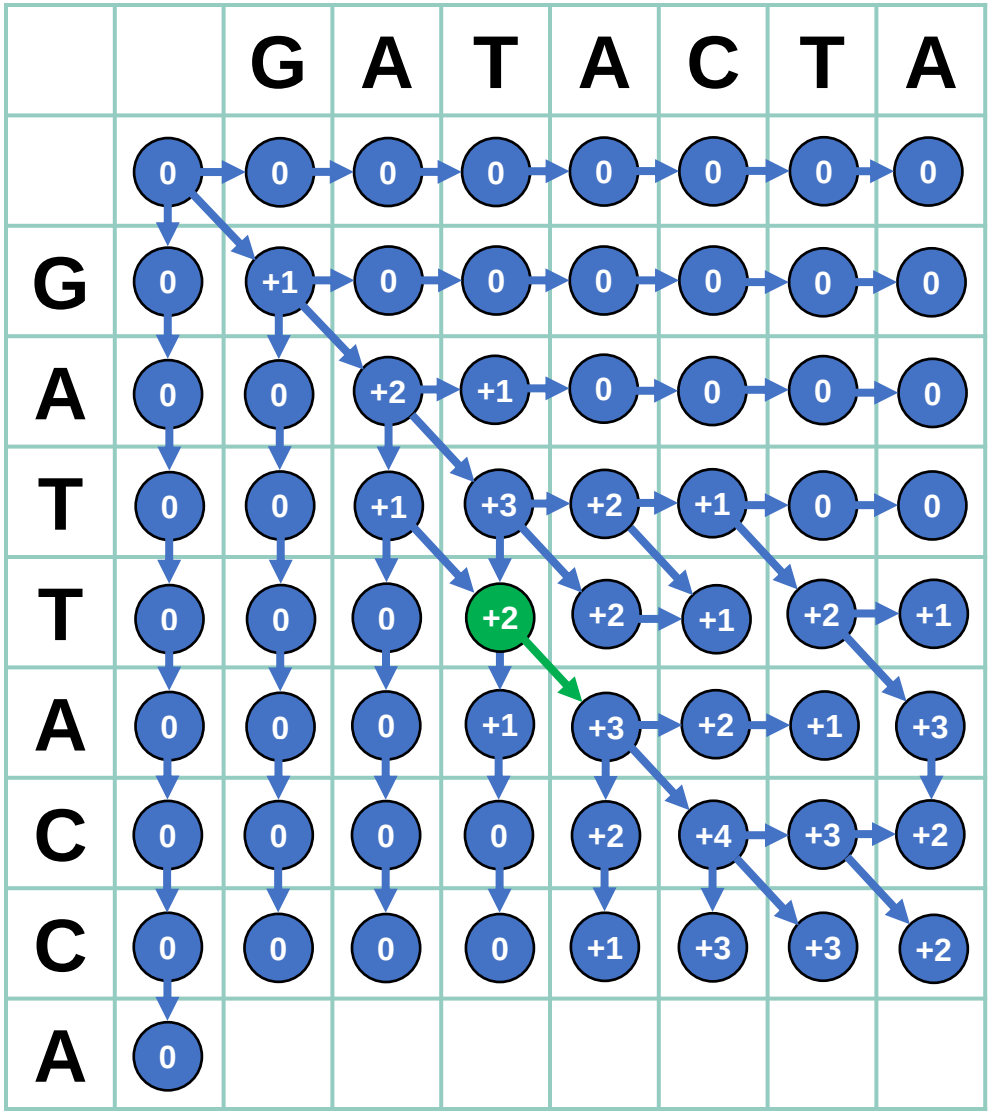
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

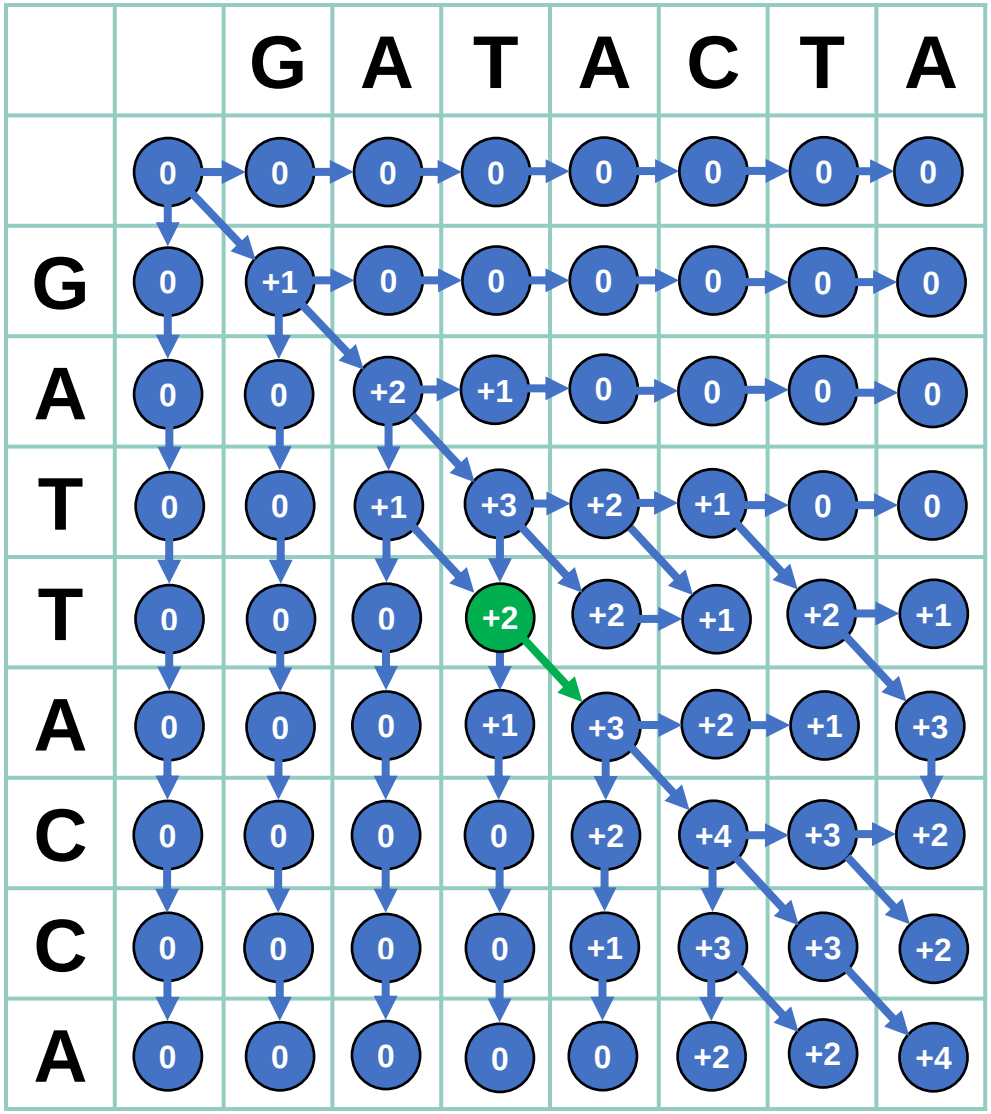
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

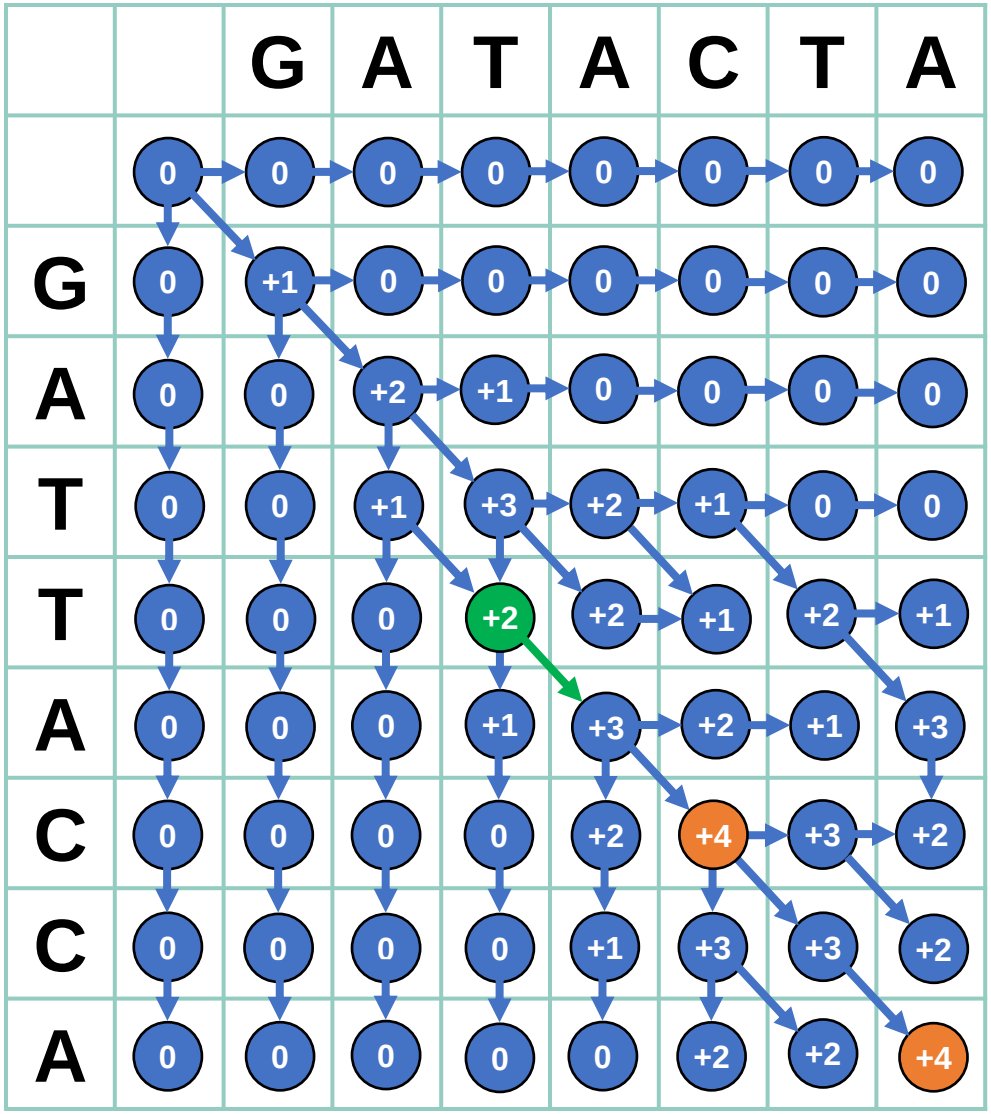
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

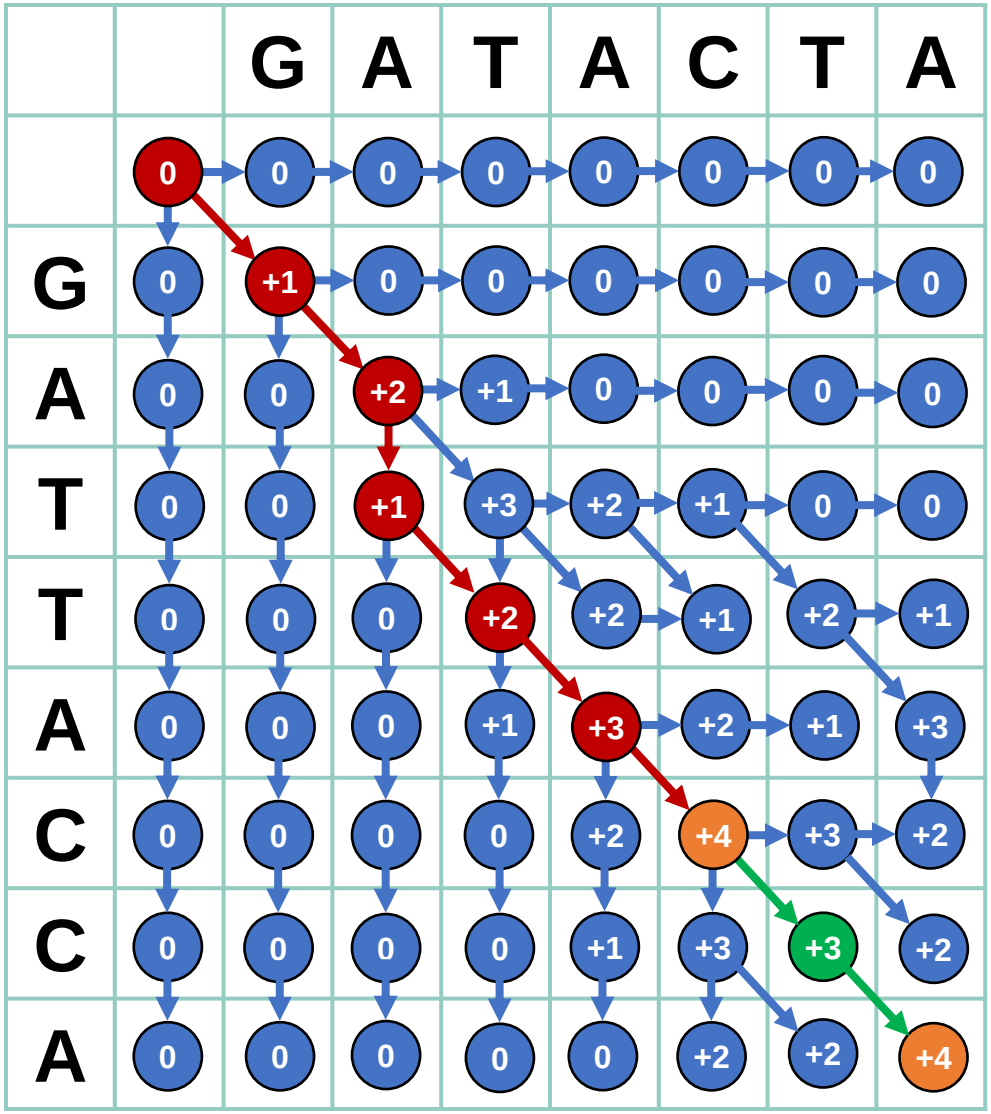
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

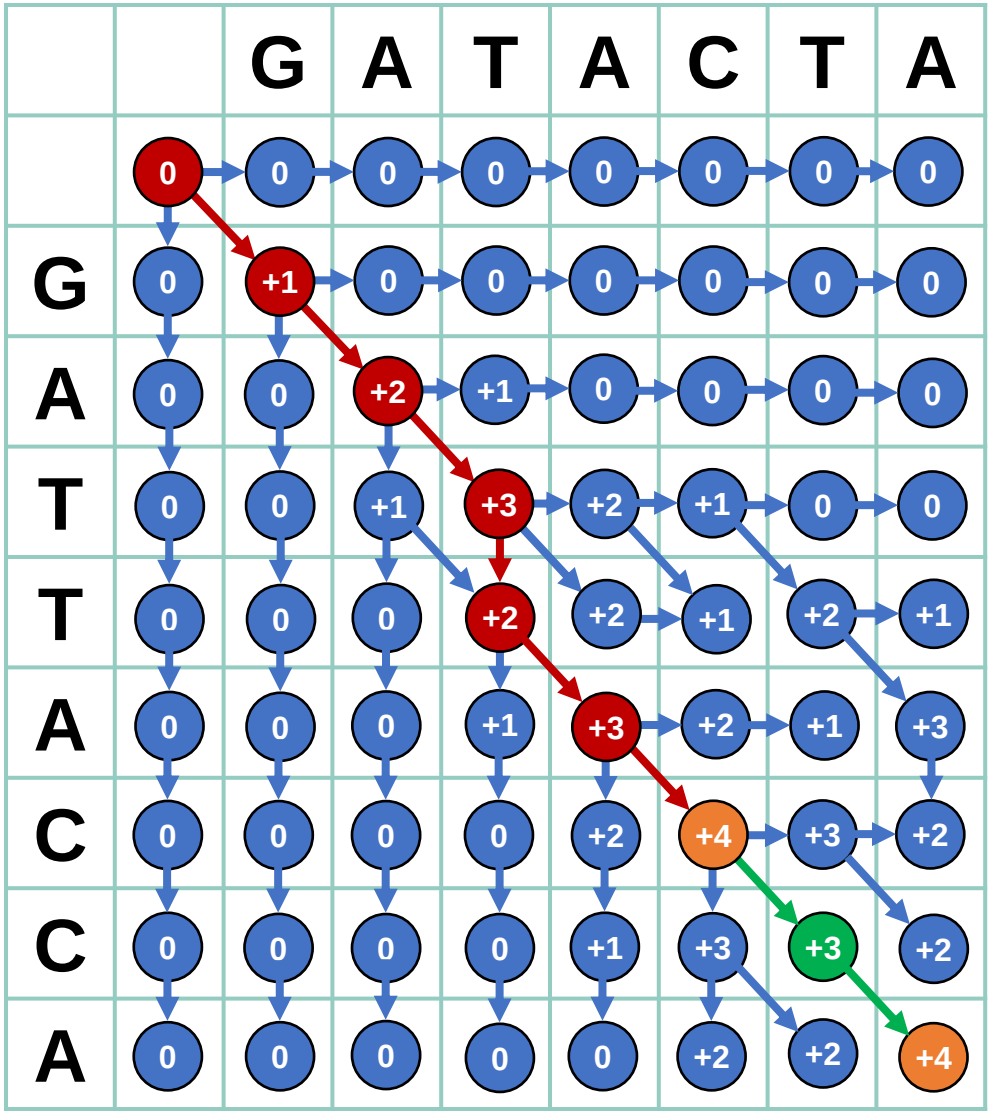
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

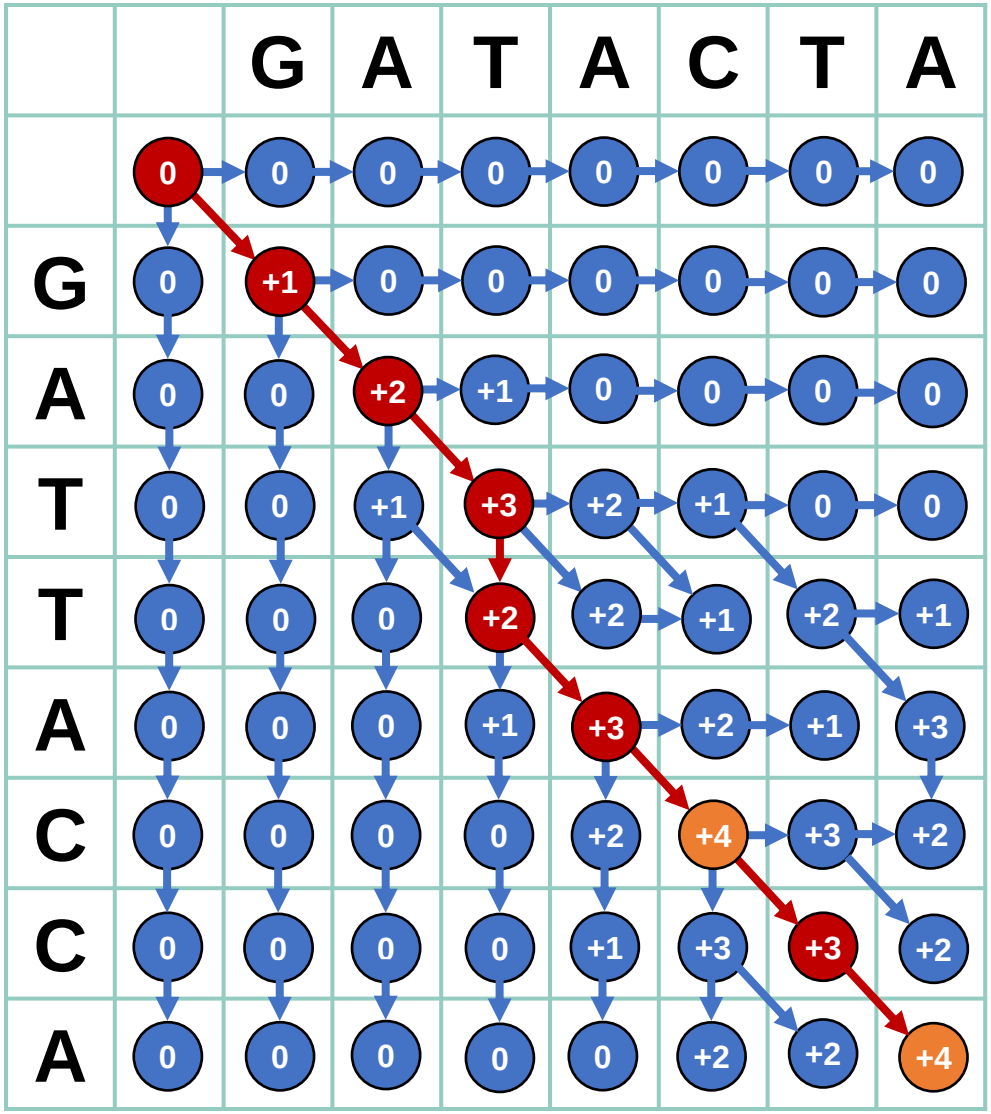
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

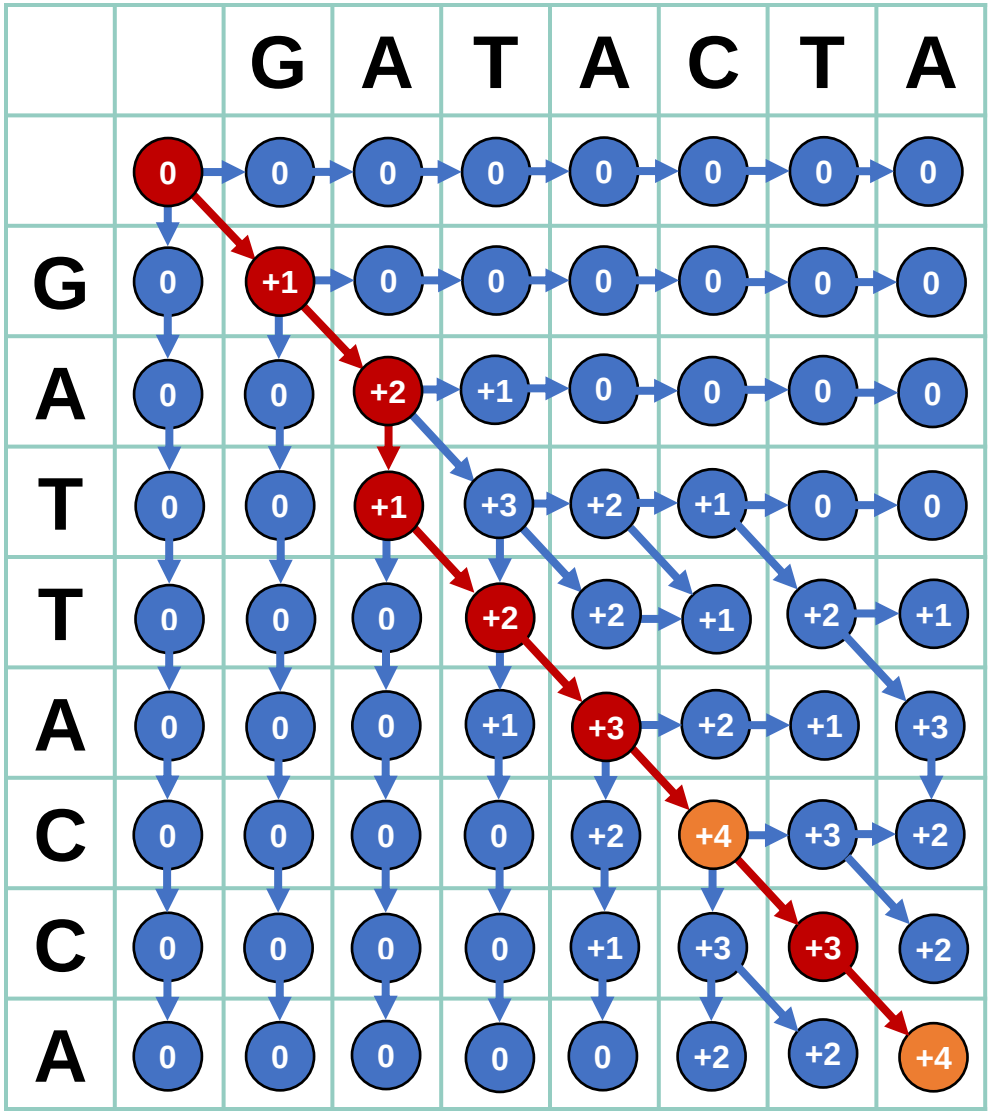
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

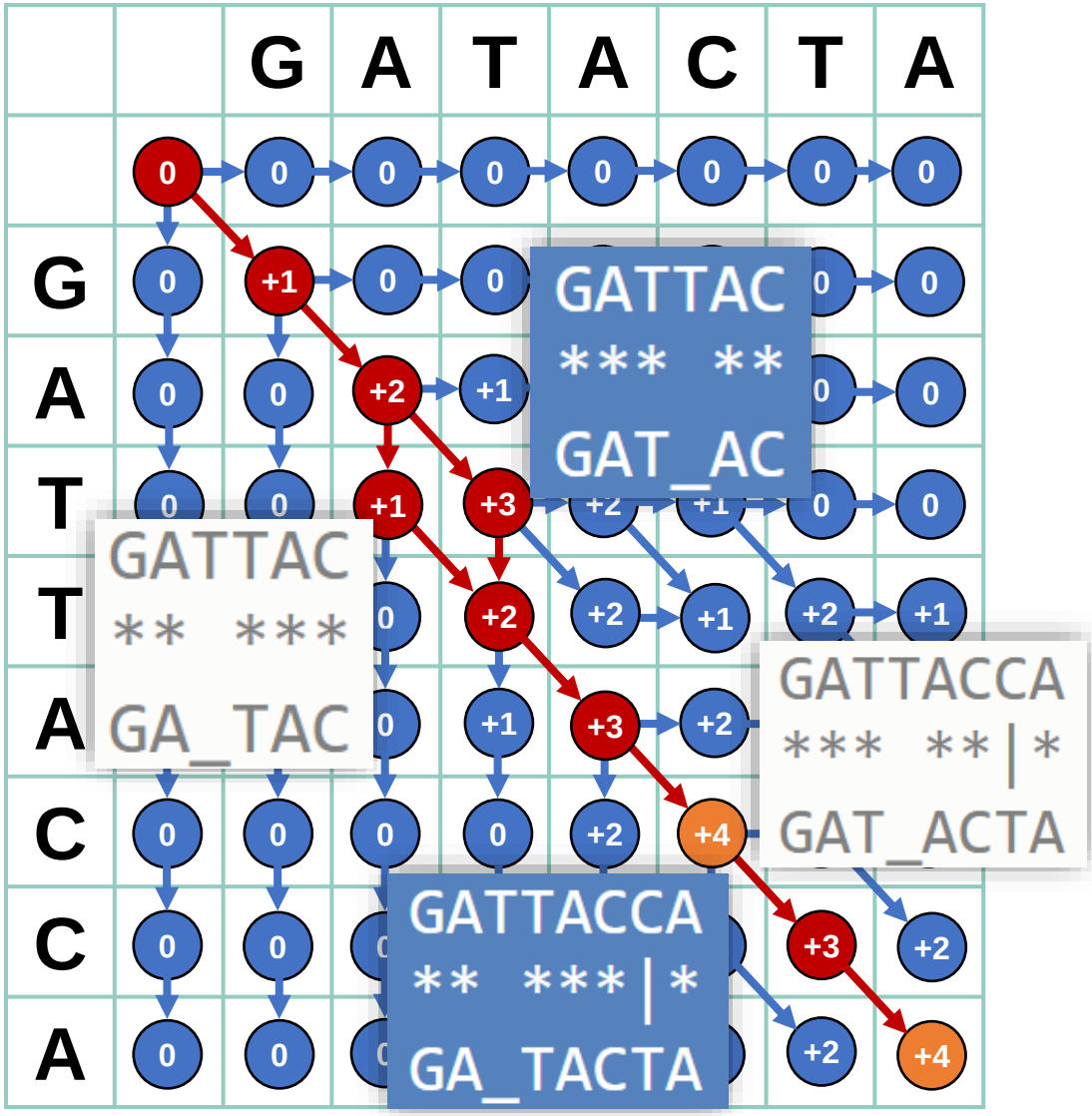
Dynamic programming

Match	Gap	Gap
G	G	-
G	-	G
+1	-1	-1

Puntajes	
Match	+1
Mismatch	-1
Gap	-1

Algoritmo de Smith & Waterman (SW)

Alineamiento Local



El score para una ruta es la suma incremental de los scores de sus pasos (diagonales o lados).

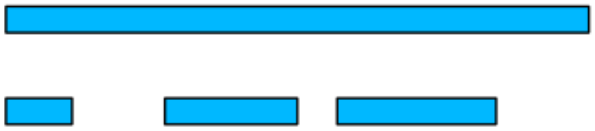
Global y local

- **Un algoritmo de alineamiento local, siempre produce alineamientos locales?**
- **Un algoritmo de alineamiento global siempre produce alineamientos globales?**

- **NO**
  - **dependiendo del sistema de scoring (scores para match/mismatch/gaps) SW puede producir alineamientos globales**
  - **dependiendo la penalidad asignada a los gaps en los extremos de un alineamiento global (o alterando significativamente el sistema de scoring) NW puede producir alineamientos locales**

Alineamiento Local

- A partir de dos secuencias, encuentra el mejor alineamiento de sub-secuencias y su puntuación.
- Se desea ignorar las regiones de las secuencias cuyo alineamiento es malo.
- Se utilizan el mismo tipo de matrices de sustitución y reglas de puntuación de gaps.
- Utiliza una modificación del método de programación dinámica de NW.



- Algoritmo de **Smith-Waterman**

Que hemos visto hasta ahora…

- Los algoritmos de programación dinámica pueden resolver los alineamientos global, local y ends-free.
- Estos obtienen como resultado la puntuación y el alineamiento óptimo utilizando los parámetros dados.
- El método de dividir y conquistar hace que la complejidad espacial sea más manejable para secuencias pequeñas y medianas.

Aspectos de Programación Dinámica

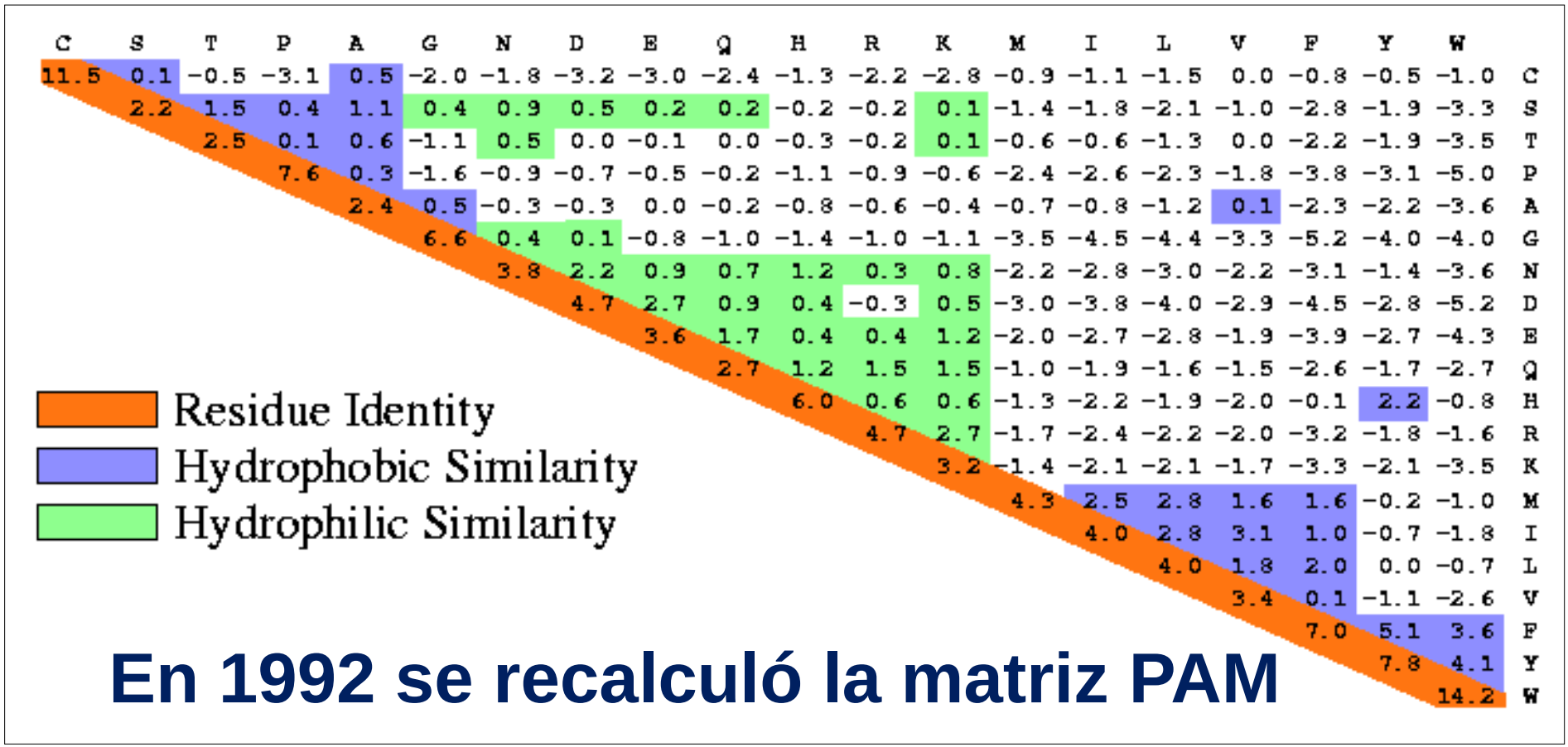
- Cuando las secuencias son enormes, hasta las limitaciones de espacio lineal son un problema.
- Aquí hemos utilizado reglas de gap muy simples.
- La reglas de gap más utilizadas son las de Afinidad:
  - Costo por abrir un gap
  - Costo por extender un gap
- Es necesario llevar y evaluar el estado del ‘gap’ en la matriz de alineamiento.

Demos:

https://bioboot.github.io/bggn213_S19/class-material/nw/
http://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Smith-Waterman
http://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Needleman-Wunsch

Matrices de Scoring

Gonnet, Cohen, Benner (1992). Exhaustive matching of the entire protein sequence database. *Science*. *256*:1443-1445.



En 1992 se recalculó la matriz PAM

Gonnet PAM250

Matrices de Scoring

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
P	-3	-1	-1	7																	P
A	0	1	0	-1	4																A
G	-3	0	-2	-2	0	6															G
N	-3	1	0	-2	-2	0	6														N
D	-3	0	-1	-1	-2	-1	1	6													D
E	-4	0	-1	-1	-1	-2	0	2	5												E
Q	-3	0	-1	-1	-1	-2	0	0	2	5											Q
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8										H
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5									R
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5								K
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5							M
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4					L
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4				V
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6			F
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7		Y
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11	W
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	

small hydrophylic

acid hydrophylic

basic

small hydrophobic

aromatic

BLOSUM 62

Heurística

Heurística

Se puede definir **Heurística** como un arte, técnica o procedimiento práctico o informal, para resolver problemas.[1] Alternativamente, se puede definir como un conjunto de reglas metodológicas no necesariamente formalizadas, positivas y negativas, que sugieren o establecen cómo proceder y problemas a evitar a la hora de generar soluciones y elaborar hipótesis.[2]

Es generalmente considerado que la capacidad heurística es un rasgo característico de los humanos[3] desde cuyo punto de vista puede describirse como *el arte y la ciencia del descubrimiento y de la invención* o de resolver problemas mediante la creatividad y el pensamiento lateral o pensamiento divergente. Según el matemático George Pólya[4] la base de la heurística está en la experiencia de resolver problemas y en ver cómo otros lo hacen. Consecuentemente se dice que hay búsquedas ciegas, búsquedas heurísticas (basadas en la experiencia) y búsquedas racionales.

problemas a evitar a la hora de generar soluciones y elaborar hipótesis.[2]

Heuristic Methods (BLAST, FASTA)

- Dynamic Programming: computational method that provide in mathematical sense the best alignment between two sequences, given a scoring system.
- Heuristic Methods (e.g. BLAST, FASTA) they prune the search space by using fast approximate methods to select the sequences of the database that are likely to be similar to the query and to locate the similarity region inside them
  - Restricting the alignment process:
    - Only to the selected sequences
    - Only to some portions of the sequences (search as small a fraction as possible of the cells in the dynamic programming matrix)

Two methods that are at least 50-100 times faster than dynamic programming were developed: FASTA and BLAST

- These methods are heuristic; i.e., an empirical method of computer programming in which rules of thumb are used to find solutions.
- They almost always works to find related sequences in a database search but does not have the underlying guarantee of an optimal solution like the dynamic programming algorithm.

La PD no es adecuada para buscar en BD

Heuristic Methods (BLAST, FASTA)

Es el método adecuado para la búsqueda en bases de datos

- **Se escoge un sistema de puntuación (matriz, costes, …) y se alinea la secuencia con cada una de las que contiene la base de datos.**
- **Se ordenan las secuencias de mayor a menor puntuación.**
- **Las secuencias de la BD con mayor puntuación y un mínimo grado de similaridad/identidad son:**

Similares a la secuencia problema

Candidatas a ser _homólogas_

El método sugerido presenta algunas dificultades

- ¿Cuánto es “un mínimo de similaridad”?
- ¿Cómo se obtiene un método que seleccione el máximo de homólogos reales y descarte las similaridades no debidas a homología?
- Que sea rapido para buscar en la una base datos!!
- Que algoritmo es el más adecuado para realizar los alineamientos?
- Programacion dinamica??? (PD)

Suposiciones en que se basa la búsqueda en las BD

Similar a lo que ya vimos en la comparación de secuencias

- La búsqueda en una base de datos presupone que…
  - **Las secuencias buscadas tienen ancestros comunes con la secuencia problema.**
  - El camino evolutivo más adecuado es el que presupone un menor número de cambios.
    - No todas las sustituciones son igualmente probables: Debemos usar matrices de sustitución que las ponderen adecuadamente.
    - Las inserciones y eliminaciones son menos probables que las sustituciones

¿Y si no esta una secuencia con ancestria en comun? ¿como me doy cuenta?

Heuristic Methods (BLAST, FASTA)

- La elección del algoritmo de búsqueda influye en la búsqueda:
  - La sensibilidad
  - La especificidad
- La elección de la matriz (tabla) de puntuación determina el patrón y la cantidad supuesta de sustituciones en las secuencias que se espera descubrir en la búsqueda.

Sensibilidad vs especificidad

El sacrificio por obtener un método mas rápido

Programacion dinámica:

- proporciona una gran sensibilidad
- Es poco específica → Pocos falsos negativos mas falsos positivos...¿?)
- Es necesariamente lenta

Métodos heurísticos

- Aproximaciones a SW con restricciones o reglas que:
- Aumentan la especificidad (aunque baja la sensibilidad)
- Son mucho más rápidas

Un poco de historia

1985 : FASTP (D. Lipman and W. Pearson)

Global gapped alignments

1988 : FASTA (W. Pearson and D. Lipman)

Local gapped alignments

1990 : BLAST1

(S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman)

Local ungapped alignments

Gapped BLASTs :

1996: WU–BLAST2 (W. Gish)

1997: NCBI–BLAST2 (and PSI–BLAST)

(S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller and D. Lipman)

El algoritmo FASTA

- **FASTA is a multistep algorithm for sequence alignment**
- **Main idea:**
  - Choose regions of the two sequences that look promising (have some degree of similarity)
  - Compute local alignment using dynamic programming in these regions

FASTP and FASTA achieve much of their speed and selectivity in the first step, by using a lookup table to locate all identities or groups of identities between two DNA or amino acid sequences during the first step of the comparison (2). The *ktup* parameter determines how many consecutive identities are required in a match. For example, if *ktup* = 4 for a DNA sequence comparison, only those identities that occur in a run of four consecutive matches are examined. In the first step, the 10 best diagonal regions are found using a simple formula based on the number of *ktup* matches and the distance between the matches without considering shorter runs of identities, conservative replacements, insertions, or deletions (1, 3).

El algoritmo FASTA

FASTA algorithm has five steps:

1. Identify common k-words in the two sequences (Query and target) (k=1,2 for proteins, 4-6 for DNA sequences)
2. Score diagonals with k-word matches, identify 10 best diagonals
3. *Rescore initial regions with a substitution score matrix*
4. *Join initial regions using gaps, penalise for gaps*
5. Perform dynamic programming to find final alignments using the optimized score

Etapa nº 1: Identidad

En la primera etapa solo se fija en los residuos idénticos entre las 2 secuencias

El proceso descrito a continuación se aplica a cada una de las secuencias de la base de datos cuando se compara con la secuencia de consulta**: los autores llaman "k - tupla" a una palabra, o fragmento de secuencia de longitud** k; todas las k - tuplas de la secuencia de consulta (obtenidas por deslizamientos sucesivos de una ventana de longitud k a lo largo de la secuencia) se comparan con todas las de la secuencia de la base de datos y las identidades entre k - tuplas se detectan y almacenan.
El método puede representarse en una matriz bidimensional, y cada identidad entre dos k - tuplas se representa como un punto.

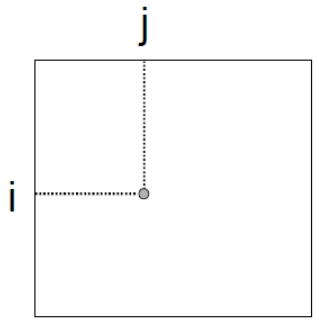
Cada segmento diagonal corresponde a una alineación sin huecos de las regiones de las dos secuencias, como una sucesión de coincidencias y desajustes de tuplas k.

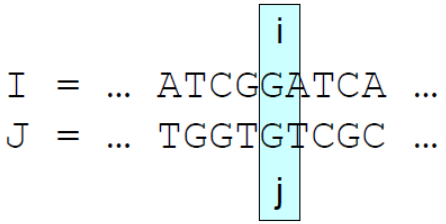
Word lists and comparison by content l The k-words of I can be arranged into a table of word occurences Lw(I)
l Consider the k-words when k=2 and I=GCATCGGC:

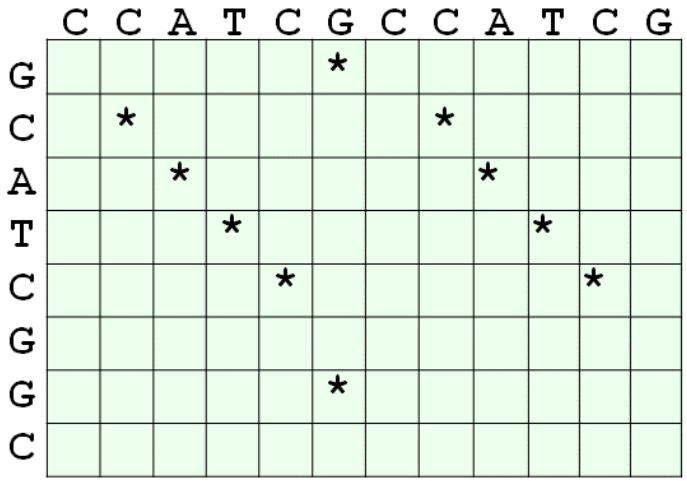
GC, CA, AT, TC, CG, GG, GC
AT: 3
CA: 2
CG: 5
GC: 1, 7
GG: 6
TC: 4
Start indecies of k-word GC in I
Building Lw(I) takes O(n) time

Dot matrix comparisons

- Word matches in two sequences I and J can be represented as a *dot matrix*
- Dot matrix element (i, j) has ”a dot”, if the word starting at position i in I is identical to the word starting at position j in J
- The dot matrix can be plotted for various k







The value chosen for the k parameter is important: the smaller k is, the greater the sensitivity, for example, if $k = 1$, one will take into account all the matches between individual amino acids, else, one will consider only stretches of k successive identities and then be more stringent. The greater k is, the faster the whole program will be, because less regions will be considered in the next steps. The choice of k will then reflects a trade-off between sensitivity and speed issues.

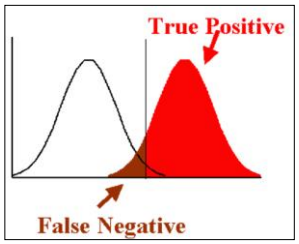
Menor *k* = Mayor sensibilidad = Menor especificidad = menos FN = más FP

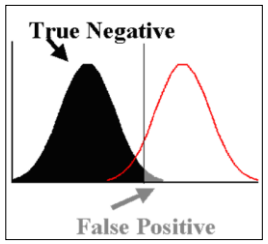
Menor *k* = Menor velocidad de computación = Mayor necesidad de memoria del ordenador

Mayor *k* = Menor sensibilidad = Mayor especificidad = más FN = menos FP

Mayor *k* = Mayor velocidad de computación y menor necesidad de memoria del ordenador

La importancia del valor de *k*

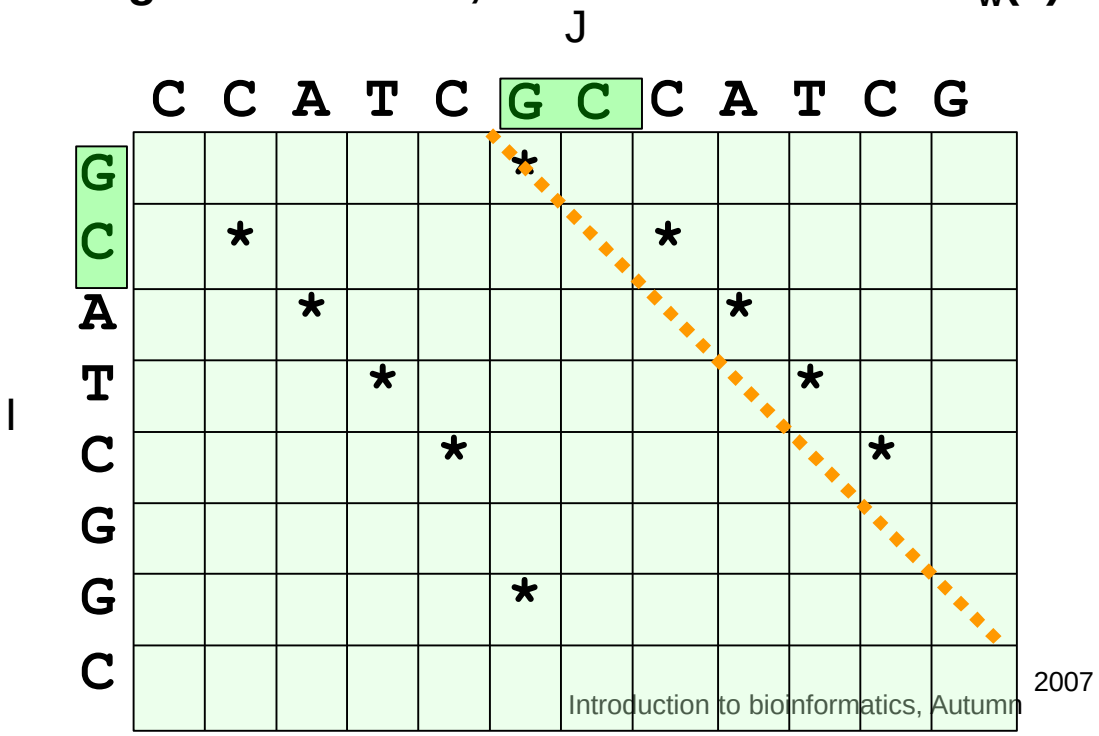




FASTA algorithm has five steps:

1. Identify common k-words in the two sequences (Query and target) (k=1,2 for proteins, 4-6 for DNA sequences)
2. Score diagonals with k-word matches, identify 10 best diagonals
3. *Rescore initial regions with a substitution score matrix*
4. *Join initial regions using gaps, penalise for gaps*
5. Perform dynamic programming to find final alignments using the optimized score

Go through k-words of I, look for matches in $L_w(J)$ and update diagonal sums

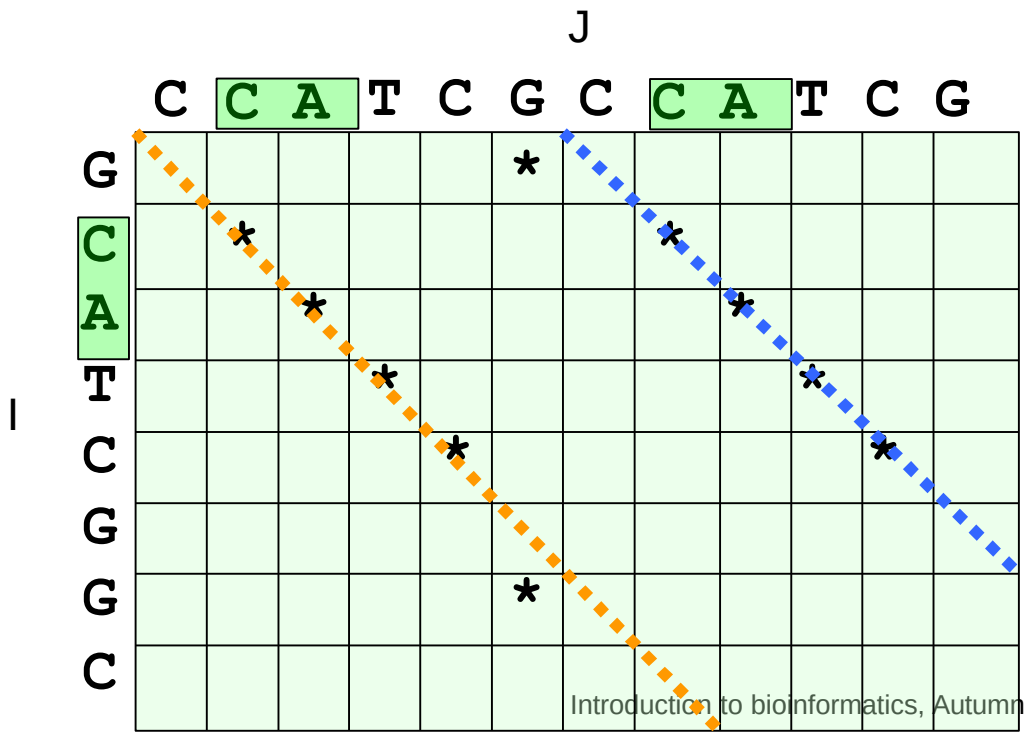


For the first 2-word in I, GC, $L_{GC}(J) = \{6\}$.

We can then update the sum of diagonal $l = i - j = 1 - 6 = -5$ to $S_{-5} := S_{-5} + 1 = 0 + 1 = 1$

Computing diagonal sums

Go through k-words of I, look for matches in $L_w(J)$ and update diagonal sums



Next 2-word in I is CA, for which $L_{CA}(J) = \{2, 8\}$.

Two diagonal sums are updated:

$I = i - j = 2 - 2 = 0$

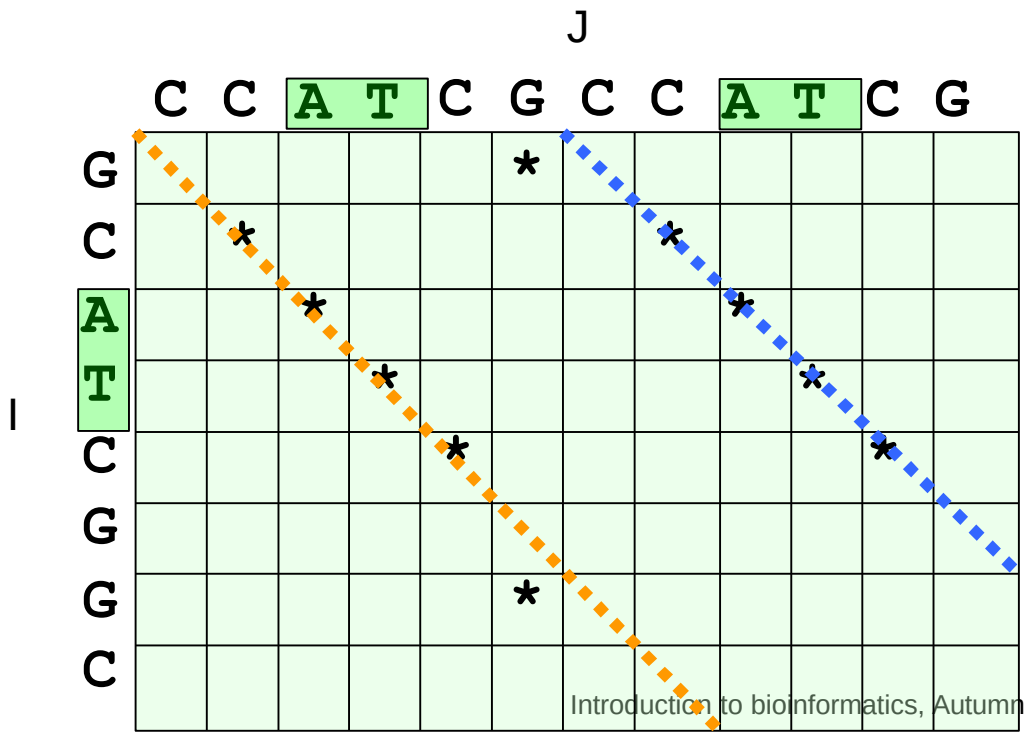
$S_0 := S_0 + 1 = 0 + 1 = 1$

$I = i - j = 2 - 8 = -6$

$S_{-6} := S_{-6} + 1 = 0 + 1 = 1$

Computing diagonal sums

Go through k-words of I, look for matches in $L_w(J)$ and update diagonal sums



Next 2-word in I is AT, for which $L_{AT}(J) = \{3, 9\}$.

Two diagonal sums are updated:

I = i – j = 3 – 3 = 0

$S_0 := S_0 + 1 = 1 + 1 = 2$

I = i – j = 3 – 9 = -6

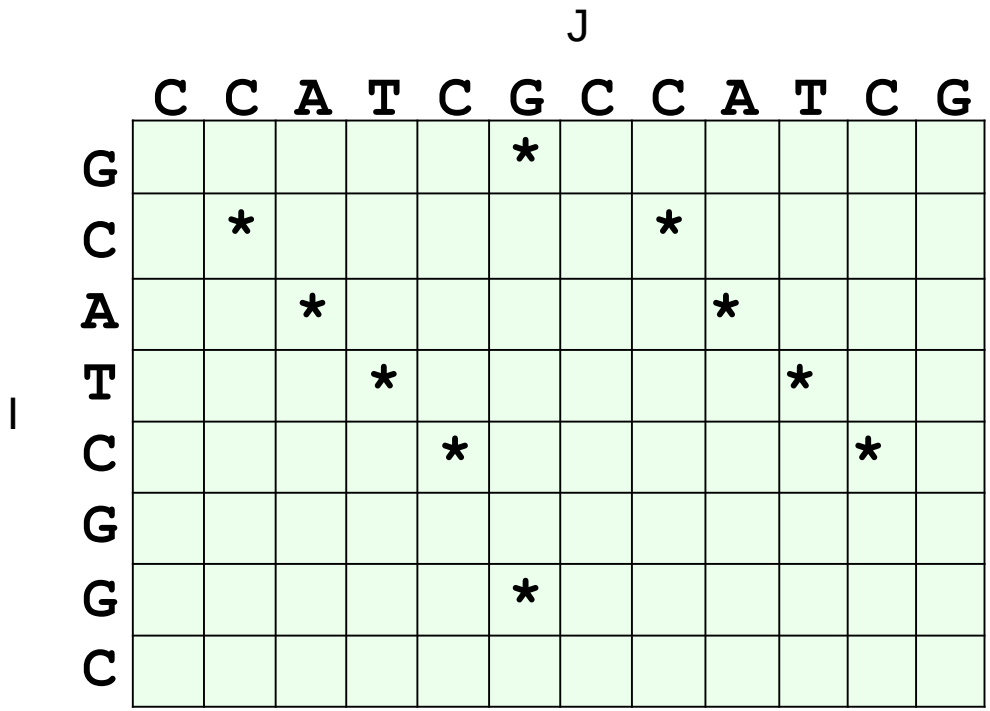
$S_{-6} := S_{-6} + 1 = 1 + 1 = 2$

2. Score diagonals with k-word matches, identify 10 best diagonals

Computing diagonal sums

After going through the k-words of I, the result is:

l	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
S_l	0	0	0	0	4	1	0	0	0	0	4	1	0	0	0	0	0

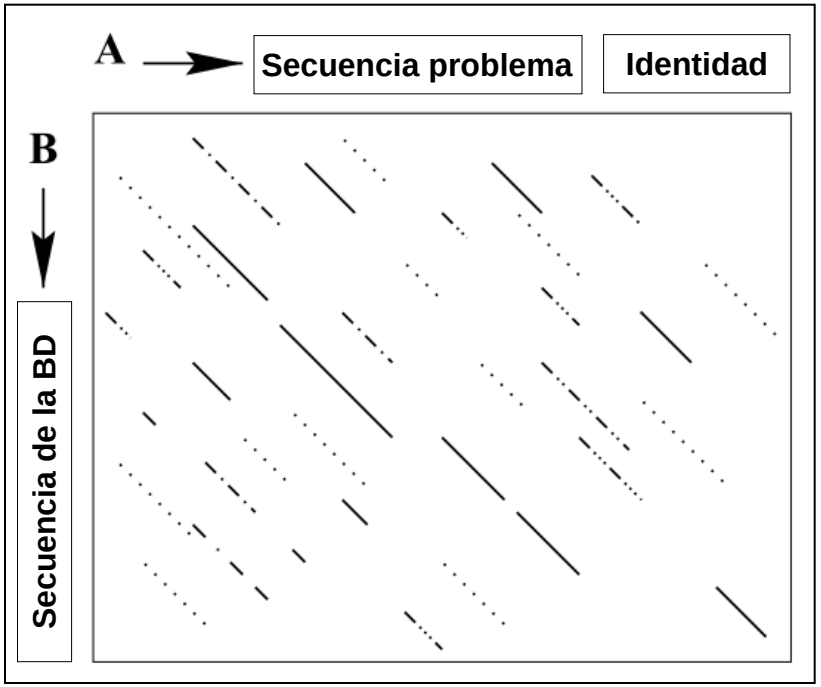


2. Score diagonals with k-word matches, identify 10 best diagonals

The second step consist of **localizing the ten best regions of similarity** between the query sequnece and each sequence of database from the best scores

A partir de una secuencia problema se obtienen todos los *k*-tuplos posibles mediante el método de la ventana deslizante.

Se compara cada *k*-tuplo con los de cada secuencia de la BD. Las regiones idénticas aparecerán como una diagonal formada por una serie de *k*-tuplos idénticos que puede verse interrumpida por fragmentos que no son idénticos.



***k* = 6 for DNA**

***k* = 2 for proteins**

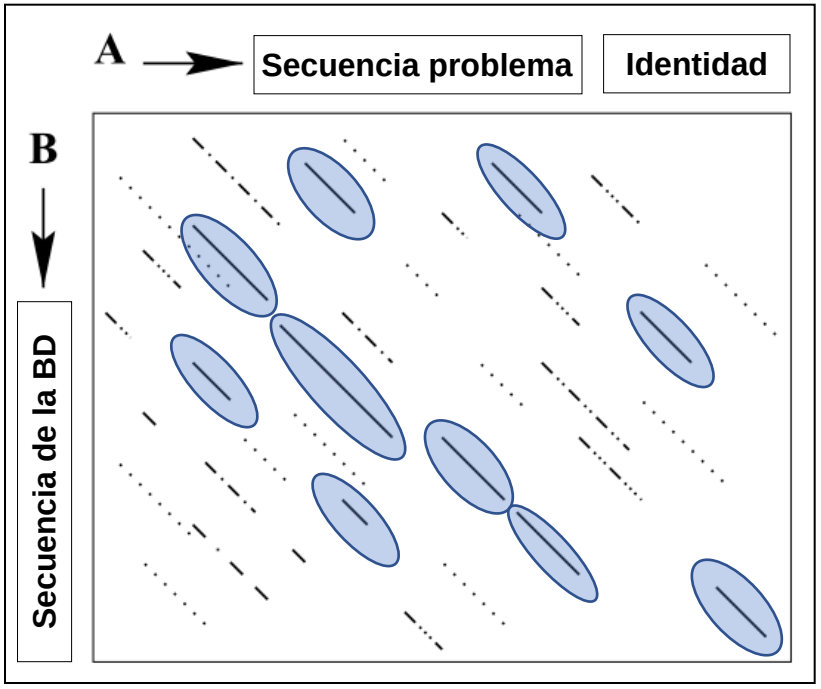
Mediante un sencillo sistema de puntuación que tiene en cuenta el número de *k*-tuplos idénticos y la separación que hay entre ellos se seleccionan las 10 diagonales que acumulan mayor densidad de *k*-tuplos idénticos.

2. Score diagonals with k-word matches, identify 10 best diagonals

The second step consist of **localizing the ten best regions of similarity** between the query sequnece and each sequence of database from the best scores

A partir de una secuencia problema se obtienen todos los *k*-tuplos posibles mediante el método de la ventana deslizante.

Se compara cada *k*-tuplo con los de cada secuencia de la BD. Las regiones idénticas aparecerán como una diagonal formada por una serie de *k*-tuplos idénticos que puede verse interrumpida por fragmentos que no son idénticos.



***k* = 6 for DNA**

***k* = 2 for proteins**

Mediante un sencillo sistema de puntuación que tiene en cuenta el número de *k*-tuplos idénticos y la separación que hay entre ellos se seleccionan las 10 diagonales que acumulan mayor densidad de *k*-tuplos idénticos.

Únicamente estas 10 regiones pasan a la siguiente etapa del método FASTA.

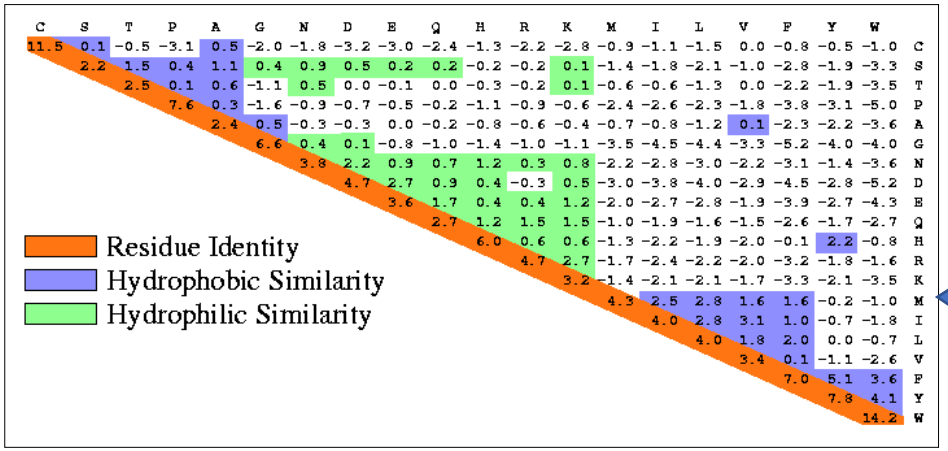
3. Rescore initial regions with a substitution score matrix

FASTA algorithm has five steps:

1. Identify common k-words in the two sequences (Query and target) (k=1,2 for proteins, 4-6 for DNA sequences)
2. Score diagonals with k-word matches, identify 10 best diagonals
3. ***Rescore initial regions with a substitution score matrix***
4. *Join initial regions using gaps, penalise for gaps*
5. Perform dynamic programming to find final alignments using the optimized score

En la segunda etapa (limitada a los regiones *top ten*) también se tienen en cuenta los residuos similares entre las 2 secuencias

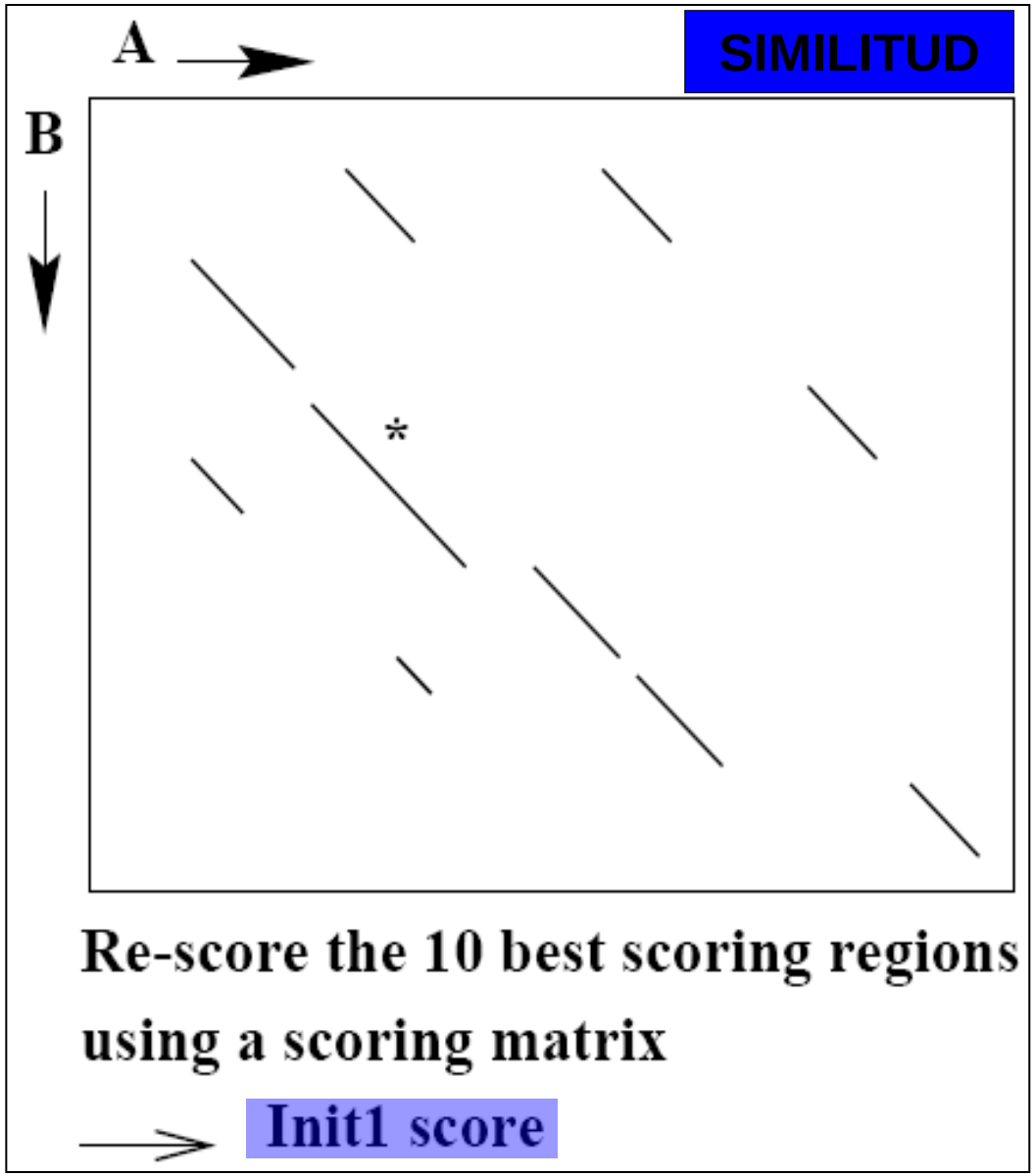
we rescore these 10 regions using a scoring matrix that allows conservative replacements and runs of identities shorter than *ktup* to contribute to the similarity score. For protein sequences, this score is usually calculated using the PAM250 matrix (4), although scoring matrices based on the minimum number of base changes required for a replacement or on an alternative measure of similarity can also be used with FASTA. For each of these best diagonal regions, a subregion with maximal score is identified. We will refer to this region as the "initial region";



Uses a predetermined scoring matrix:

- Nucleotides: Identity, Cantor-Jukes, Kimura
- Proteins: PAM250, Blosum62, Blosum50

3. Re-score initial regions with a substitution score matrix



Las 10 diagonales con mayor densidad de *k*-tuplos idénticos seleccionadas en la etapa anterior se vuelven a puntuar, esta vez utilizando una matriz de sustitución. En esta etapa también se tienen en cuenta *k*-tuplos de menor tamaño y residuos similares.

Las regiones que alcanzan mayor puntuación se denominan regiones iniciales y su puntuación se denomina *init1*.

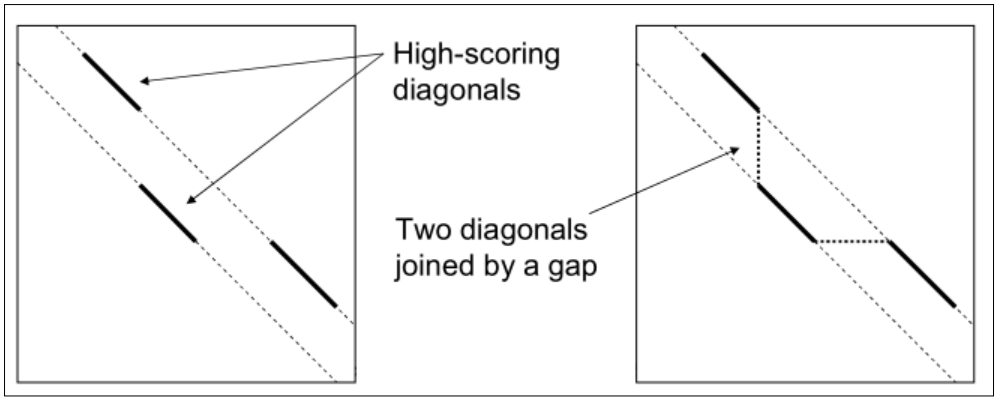
La región inicial con mayor *init1* aparece marcada con un asterisco.

- **Each high-scoring diagonal chosen in the previous step is rescored according to a score matrix**
- **This is done to find subregions with identities shorter than k**
- **Non-matching ends of the diagonal are trimmed**

FASTA algorithm has five steps:

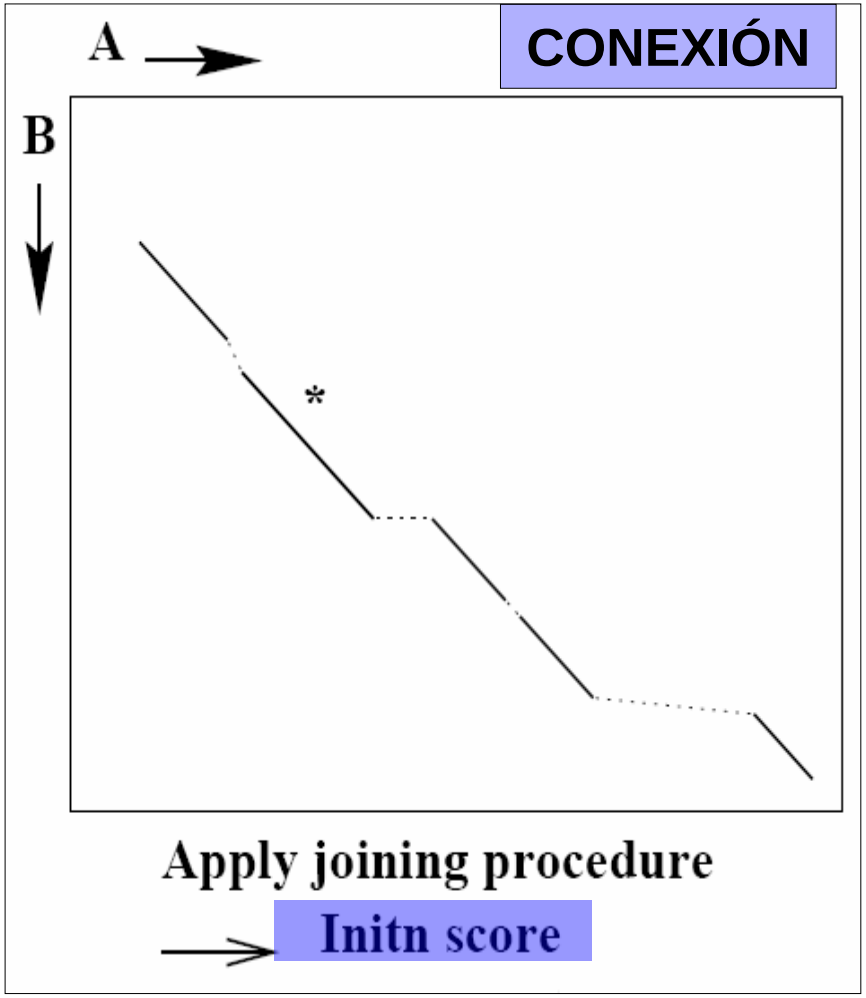
1. Identify common k-words in the two sequences (Query and target) (k=1,2 for proteins, 4-6 for DNA sequences)
2. Score diagonals with k-word matches, identify 10 best diagonals
3. *Rescore initial regions with a substitution score matrix*
4. ***Join initial regions using gaps, penalise for gaps***
5. Perform dynamic programming to find final alignments using the optimized score

FASTA goes one step further during a library search; it checks to see whether several initial regions may be joined together. Given the locations of the initial regions, their respective scores, and a ''joining'' penalty (analogous to a gap penalty), FASTA calculates an optimal alignment of initial regions as a combination of compatible regions with maximal score. FASTA uses the resulting score to rank the library sequences. We limit the degradation of selectivity by including in the optimization step only those initial regions whose scores are above a threshold.



Etapa nº 4: Conexión de regiones iniciales (se introducen huecos)

4. *Join initial regions using gaps, penalize for gaps*



FASTA intenta conectar las regiones iniciales cuya puntuación supera un determinado valor (*cutoff*).

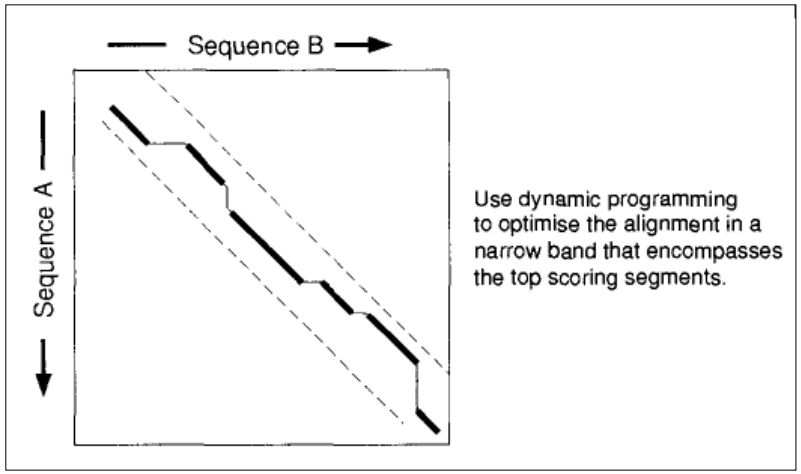
Se vuelven a puntuar las regiones unidas penalizando la introducción de huecos (*gaps*). Esta puntuación se denomina *initn* y permite hacer un *ranking* con todas las secuencias de la BD.

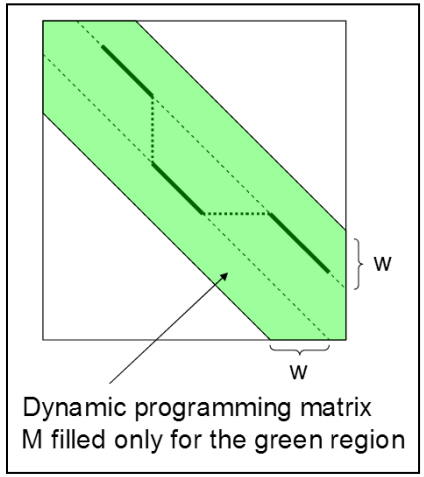
Las secuencias de la BD cuya puntuación *initn* supera un cierto umbral pasan a la quinta etapa

Etapa nº 4: puntuación *initn* y *ranking* de secuencias

FASTA algorithm has five steps:

1. Identify common k-words in the two sequences (Query and target) (k=1,2 for proteins, 4-6 for DNA sequences)
2. Score diagonals with k-word matches, identify 10 best diagonals
3. *Rescore initial regions with a substitution score matrix*
4. *Join initial regions using gaps, penalise for gaps*
5. **Perform dynamic programming to find final alignments using the optimized score**



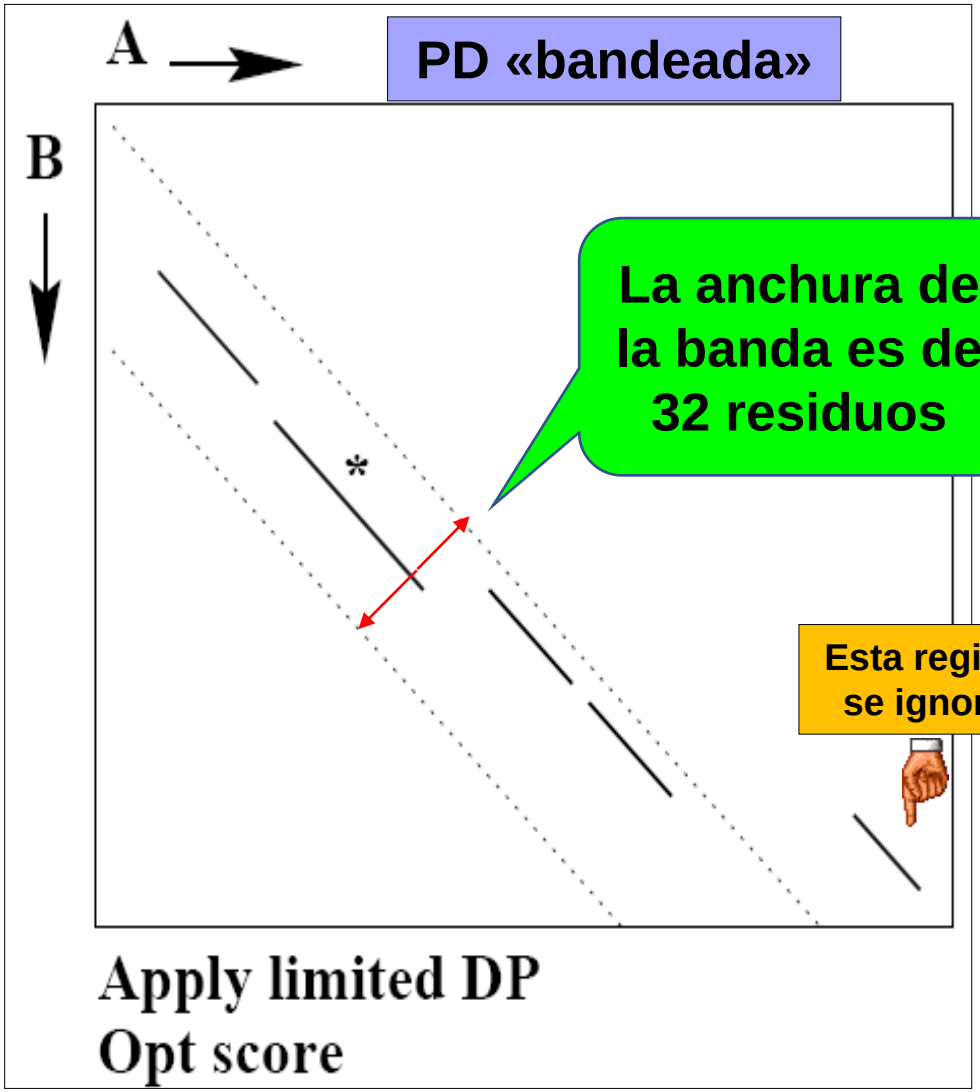


In the fourth step of the comparison, the highest scoring library sequences are aligned using a modification of the optimization method described by Needleman and Wunsch (5) and Smith and Waterman (6). This final comparison considers all possible alignments of the query and library sequence that fall within a band centered around the highest scoring initial region

Etapa nº 5: Programación dinámica “bandeada”

After a complete search of the library, FASTA plots the initial scores of each library sequence in a histogram, calculates the mean similarity score for the query sequence against each sequence in the library, and determines the standard deviation of the distribution of initial scores. The initial scores are used to rank the library sequences, and, in the fourth and final step of the comparison, the highest scoring library sequences are aligned using a modification of the standard NWS optimization method. The optimization employs the same scoring matrix used in determining the initial regions; the resulting optimized alignments are calculated for further analysis of potential relationships, and the optimized similarity score is reported.

5. Perform dynamic programming to find final alignments using the optimized score



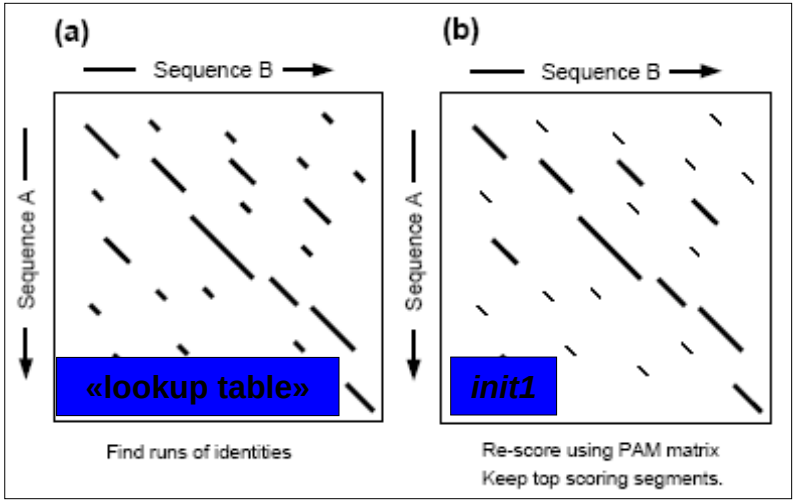
Se utiliza un algoritmo de PD modificado (SW bandeado) para alinear la secuencia problema con las secuencias de la BD. El alineamiento se limita a una estrecha banda centrada en el segmento *init1* de mayor puntuación y que contiene las diagonales con mayor puntuación.

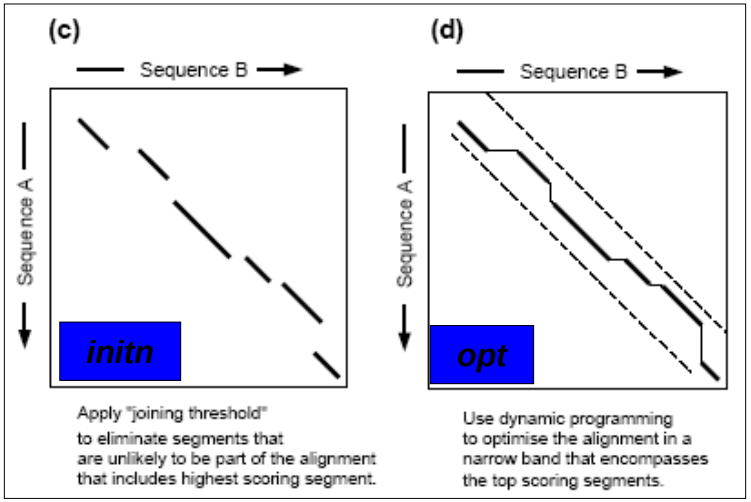
La puntuación de este alineamiento se denomina *opt*, y se utiliza para hacer un *ranking* de alineamientos. También se determina su significación estadística (denominada *E-value*).

Etapa nº 5: Alineamiento óptimo "bandeado" (*opt*)

Las 5 etapas de FASTA

	Operación	Resultado
Etapa nº 1 y 2	IDENTIDAD	Los 10 mejores
Etapa nº 3	SIMILITUD	*init1*
Etapa nº 4	CONEXIÓN	*initn*
Etapa nº 5	PD bandeada	*opt* + E-value





El algoritmo FASTA

Resumen: FASTA algorithm has five steps:

Step 1 Identify regions shared by the two sequences with the highest density of identities *(ktup=l)* or pairs of identities *(ktup=2)* between the Query and target) (k=1,2 for proteins, 4-6 for DNA sequences)

Step 2 Score diagonals with k-word matches and identify the 10 best diagonals

Step 3 Rescan the 10 regions with the highest density of identities using the PAM250 matrix. Trim the ends of the region to include only those residues contributing to the highest score. Each region is a partial alignment without gaps. For each of the best diagonal regions rescanned with the scoring matrix, a subregion with the maximal score is identified. Initial scores are used to rank the library sequences. These scores are referred to as init1 score.

Step 4 If there are several initial regions with scores greater than the CUTOFF value, check to see whether the trimmed initial regions can be joined to form an approximate alignment with gaps. Calculate a similarity score that is the sum of the joined initial regions minus a penalty (usually 20) for each gap (initn). This initial similarity score (initn) is used to rank the library sequences. The score of the single best initial region found in step 3 is reported (init1).

Step 5 Each of the selected sequences is aligned with the query sequence, using an algorithm that is a variation of the Smith-Waterman algorithm: a local alignment is calculated by Dynamic Programming, but restricting the region of the matrix that is being explored to only a band centered around the diagonal region that had got the best initl score

The **quality** of the alignment is represented by the Score (s).

The **significance** of the alignment is computed as an E- value.

E value (E)

Expectation value. The number of different alignments with scores equivalent to or better than S that are expected to occur in a database search by chance. **The lower the E value, the more significant the score**.

- The p-value is the probability of observing a given score, assuming the data is generated according to the null hypothesis.
- The E-value is the expected number of times that the given score would appear in a random database of the given size.
- One simple way to compute the E-value is to multiply the p-value times the size of the database.
- Thus, for a p-value of 0.001 and a database of 1,000,000 sequences, the corresponding E-value is $0.001 \times 1{,}000{,}000 = 1{,}000$.

La significancia del resultado

E-value II

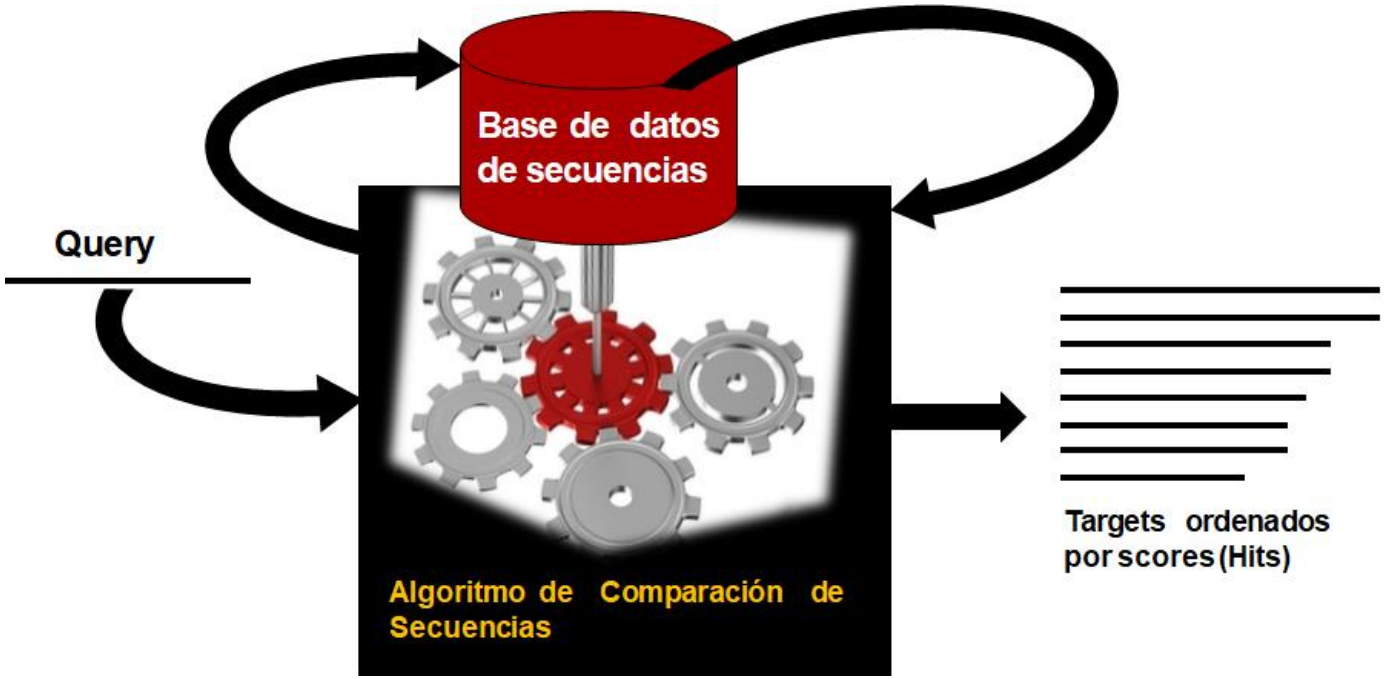
- E < 10^{-50} … extremely high confidence that the database match is a result of homologous relationships
- E is from (10^{-50}, 0.01) … the match can be considered a result of homology (**for proteins, conclusive are E-values < 0.001**)
- E is from (0.01, 10) … the match is considered not significant, but may hint tentative remote homology
- E > 10 … the sequences under consideration are either unrelated or related by extremely distant relationships that fall below the limit of detection with the current method.

- E-value is proportional to the database size, as database grows E-value for a given sequence match increases. However, the evolutionary relationship between two sequences remains constant. As the db grows, one may lose previously detected homologs.

Valor E y significancia del resultado

BLAST

(Basic Local Alignment Search Tool)



BLAST: Es un conjunto de programas para buscar similitud de secuencias en las bases de datos del NCBI o personalizada que permite la búsqueda rápida. El programa busca principalmente alineamientos locales y es capaz de detectar relaciones entre secuencias aún distantemente relacionadas.

Actualmente es el programa más popular para el análisis de secuencias.

BLAST

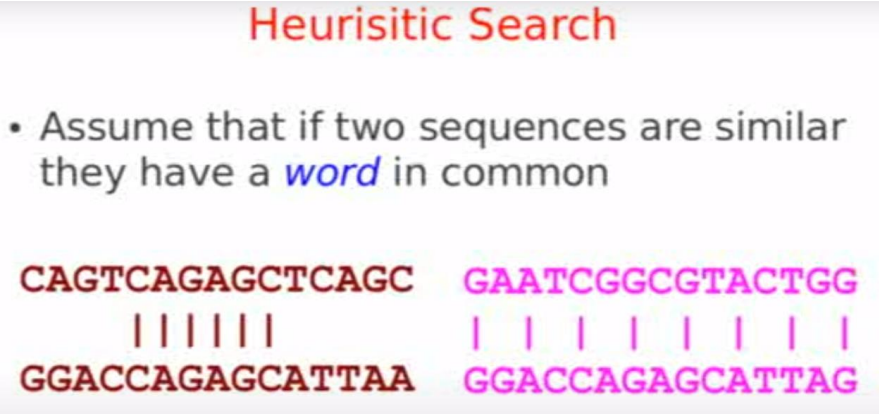
Some BLAST terminology:

Word: substring of a sequence
word pair: pair of words of the same length.
score of a word pair: score of the gapless alignment of the two words:

V A L M R
V A K N S

Score=-4+3+-4+-3+-1 = -9 (PAM120)

HSP: high scoring sequence pair.

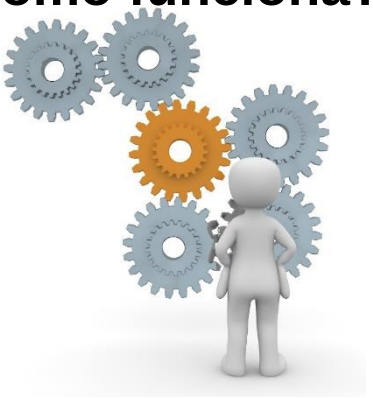


X Ej: comparten 6 letras en común vs No comparten ninguna palabra en común

BLAST puede detectar las coincidencias de la izquierda y perder o no detectar las de la derecha

BLAST

¿Como funciona?



1►Indexacion

2►Alineamiento Local

3►Extension

4►Significancia estadística del alineamiento

Main steps of BLAST:

Parameters: w = length of a hit; T = min. score of a hit (for proteins: w=3, T=13 (BLOSUM62)

Step 1: Given query sequence Q, compile the list of possible words which form with words in Q high scoring word pairs.

Step 2: Scan database for exact matching with the list of words complied in step 1.

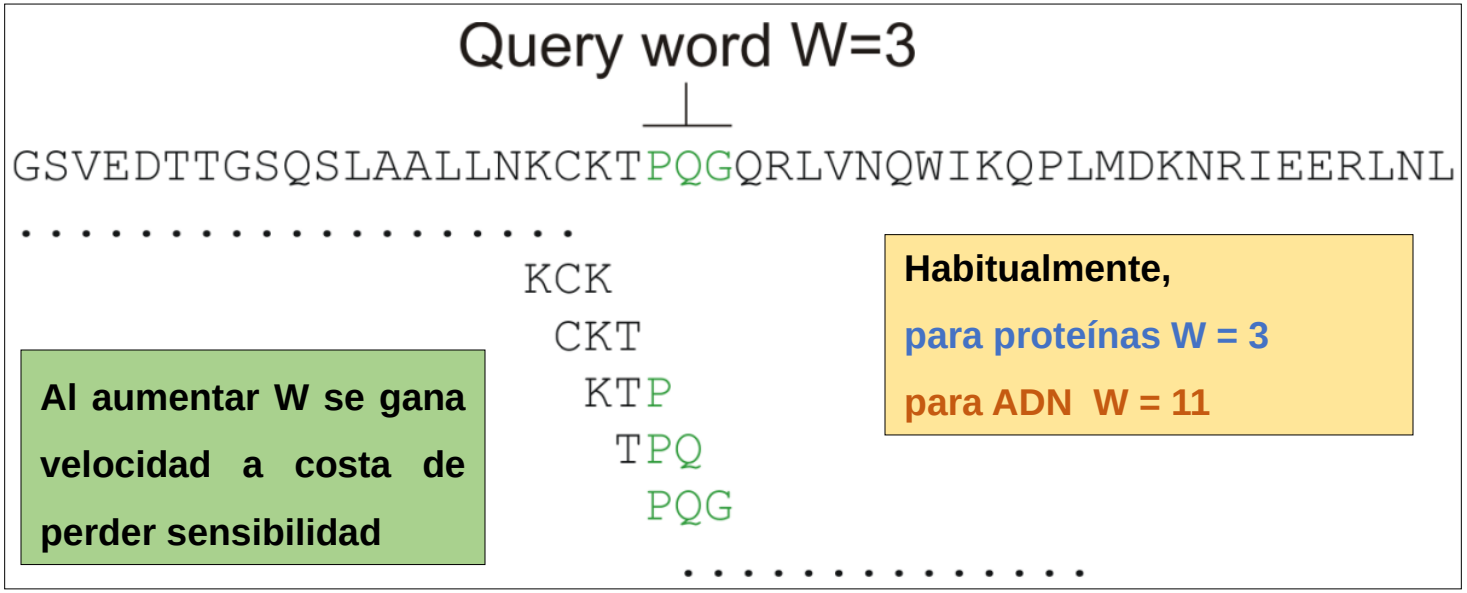
Step 3: Extending hits from step 2.

Step 4: Evaluating significance of extended hits from step 3.

Step 1: Given query sequence Q, compile the list of possible words which form with words in Q high scoring word pairs.

Se descompone la secuencia problema (Query) en "palabras"

Mediante el método de la "ventana deslizante" se descompone la secuencia problema en "palabras". El parámetro W (*word size*) determina el número de caracteres de las palabras.



Se generan un índice a cada *word* de la secuencia *Query* y de todas las secuencias *target* de la base de datos (estos ultimos ya estan calculados)

Se generan un índice a cada *word* de la secuencia *Query* y de todas las secuencias *target* de la base de datos (estos ultimos ya estan calculados)

Para que sea más rápida la búsqueda se transforma la secuencia de letras a números

Letter	Number
A	0
C	1
G	2
T	3

TGAC → 3 2 0 1

$3x4^3 + 2x4^2 + 0x4^1 + 1x4^0$

Key = 225

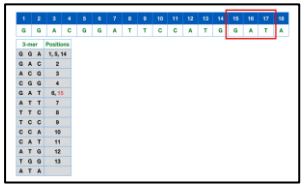
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
G	G	A	C	G	G	A	T	T	C	C	A	T	G	G	A	T	A

3-mer	Positions
G G A	1, 5, 14
G A C	2
A C G	3
C G G	4
G A T	6, 15
A T T	7
T T C	8
T C C	9
C C A	10
C A T	11
A T G	12
T G G	13
A T A	

3-mer	Positions	Key
G G A	1, 5, 14	40
G A C	2	33
A C G	3	6
C G G	4	26
G A T	6, 15	35
A T T	7	15
T T C	8	61
T C C	9	53
C C A	10	20
C A T	11	19
A T G	12	14
T G G	13	58
A T A	16	12

Se ordena de menor a mayor por key

3-mer	Positions	Key
A C G	3	6
A T A	16	12
A T G	12	14
A T T	7	15
C A T	11	19
C C A	10	20
C G G	4	26
G A C	2	33
G A T	6, 15	35
G G A	1, 5, 14	40
T C C	9	53
T G G	13	58
T T C	8	61



código binario de Word y posición

Key	6	12	14	15	19	20	26	33	35	40	53	58	61
Position	3	16	12	7	11	10	4	2	6, 15	1, 5, 14	9	13	8

Indexación: Se construye un código binario donde cada par de números hace referencia a la palabra y a la posición de la palabra en la secuencia original permitiendo rápidamente localizar vecinos (*neighbours*) para el paso de extension

https://www.youtube.com/watch?v=wfi_KimrNQM

Step 1: Given query sequence Q, compile the list of possible words which form with words in Q high scoring word pairs.

Se puntúa cada palabra aplicando una matriz de sustitución. Sólo se tendrán en cuenta las palabras cuya puntuación supere un valor T (Threshold).

Query word W=3

GSVEDTTGSQSLAALLNKCKTPQGQRLVNQWIKQPLMDKNRIEERLNLVEAF

Neighborhood Words

PQG	18
PEG	15
PRG	14
PKG	13
PNG	13
PDG	13
PHG	13
PMG	13
PSQ	13
PQA	12
PQN	12
Etc...	

Scores from BLOSUM62 matirx

Threshold for neighborhood words T=13

Al aumentar T se gana velocidad a costa de perder sensibilidad (más FN)

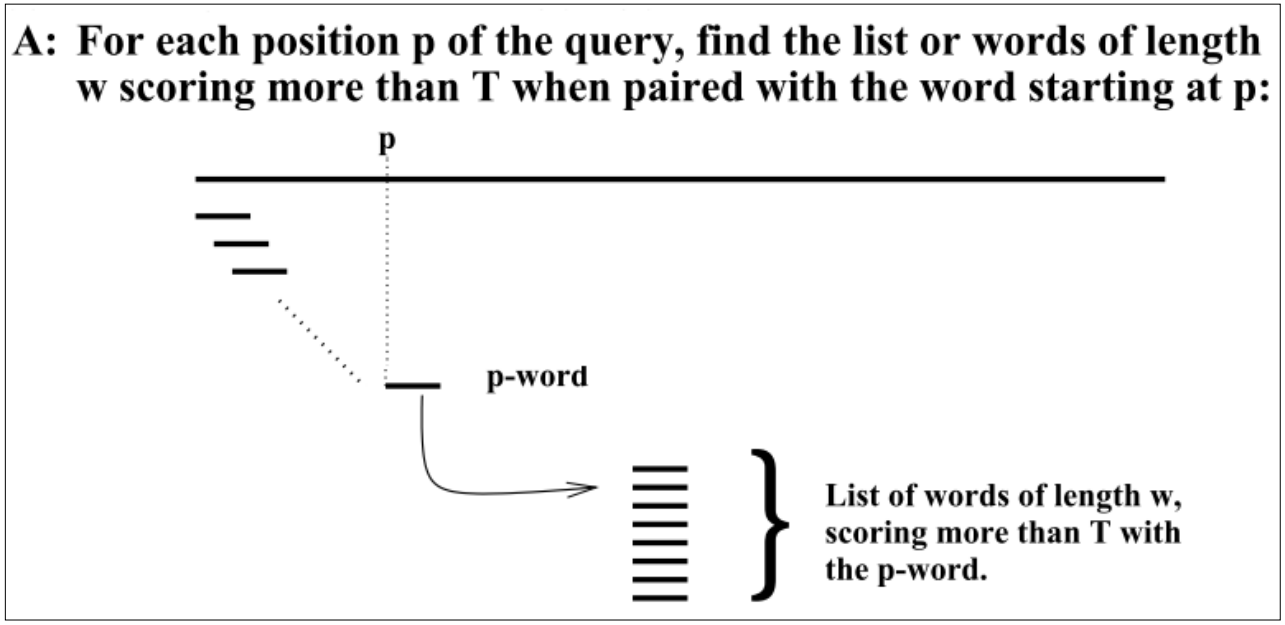
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	9																			
S	-1	4																		
T	-1	1	5																	
P	-3	-1	-1	7																
A	0	1	0	-1	4															
G	-3	0	-2	-2	0	6														
N	-3	1	0	-2	-2	0	6													
D	-3	0	-1	-1	-2	-1	1	6												
E	-4	0	-1	-1	-1	-2	0	2	5											
Q	-3	0	-1	-1	-1	-2	0	0	2	5										
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8									
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5								
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5							
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5						
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4					
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4				
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4			
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6		
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7	
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11

A cada palabra se le asocian “vecinas” (*neighbours*)

Step 1: Given query sequence Q, compile the list of possible words which form with words in Q high scoring word pairs.

Resultado de la primera etapa de BLAST:

A cada palabra de longitud w se le asocia una serie de "palabras vecinas" (*neighbours*). Cada palabra vecina recibe una puntuación (BLOSUM62) según su parecido con la palabra original. Solo se tienen en cuenta las palabras vecinas con una puntuación > *T* (*threshold*).



Tenemos una lista de Words indexada de la secuencia Query y de las secuencias de la DB de y sabemos exactamente la posición de la word en las secuencias originales y cuales son words vecinas.

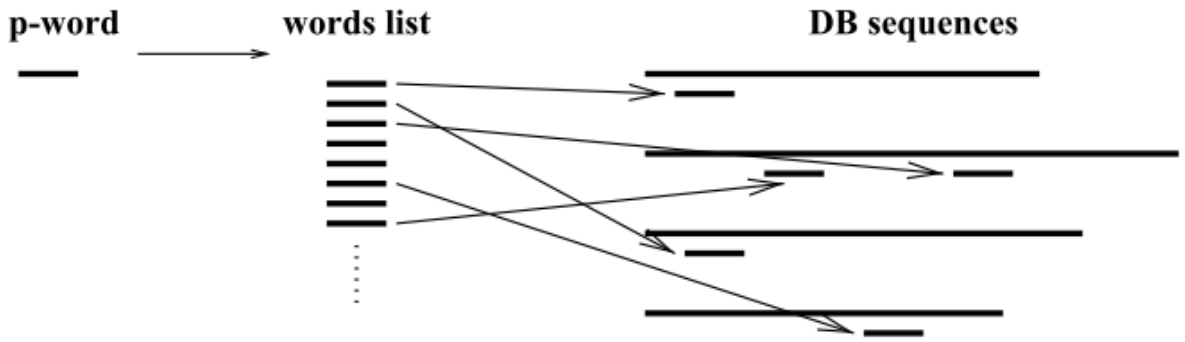
Ahora que tenemos todo ordenado y sabemos donde esta podemos iniciar la busqueda

2►Alineamiento Local

Step 2: Scan database for **exact matching** with the list of words complied in step 1.

The BLAST algorithm then starts by seeding the search with this small subset of letters from the query sequence. These words from your query sequence are then used to scan the database for matches above a certain threshold. This process of scanning a database with small sequence fragments is far faster than scanning a database with a large sequence.

B: For each words list, identify all exact matches with DB sequences:



2.1.2 Generation of *hits*

Let D be a sequence of the database, and Q be the query sequence. After the first step, Q is now represented by lists of "neighbors", one list at each position of the query. Comparing Q with D yet consists in looking for identities between the "neighbors" at each position of Q and the words of D. So, each position of Q is compared with each word of D, and if one of the neighbor words at that position of Q is identical to the word of D, a *hit* is recorded (see figure 2B). A *hit* is made with one or several successive (overlapping) pairs of similar words, and characterized by its position in each of the two sequences. All the possible *hits* between the query sequence and sequences from the database are calculated in that way.

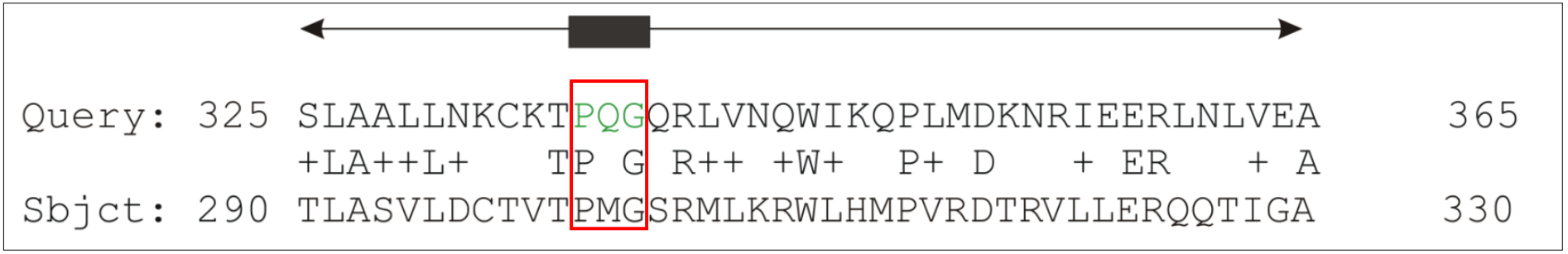
Hits o coincidencias exactas de W = 3 para proteínas y de W = 11 para ADN

(recordar que ahora tenemos numero en lugar de palabras para acelerar la búsqueda en el índice)

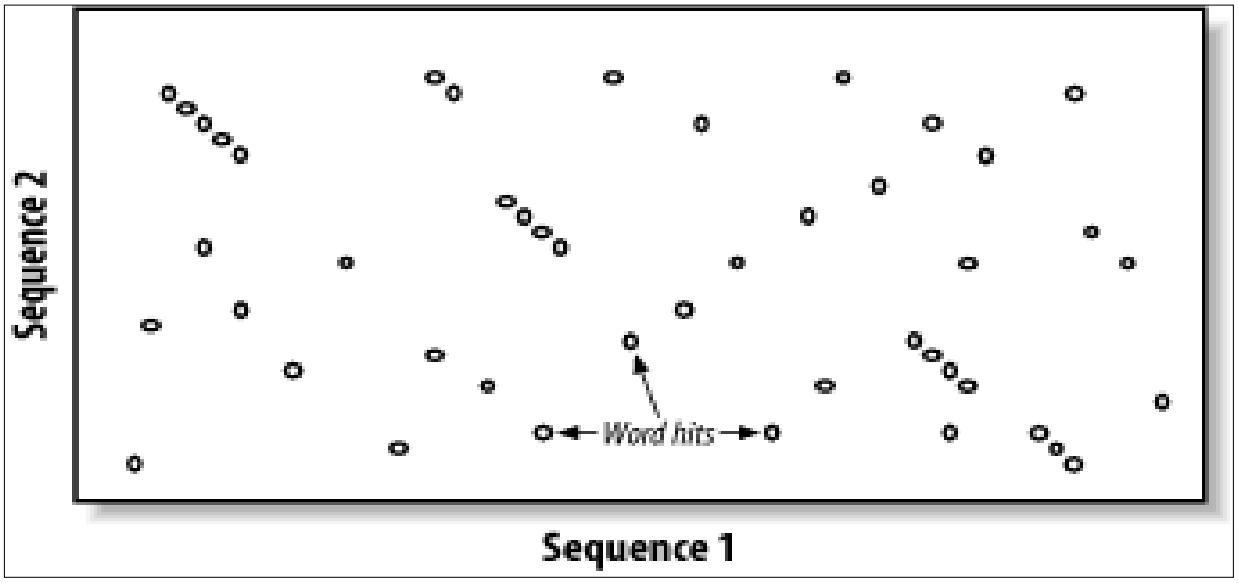
2.- Se buscan coincidencias en las secuencias de la BD

2►Alineamiento Local

Step 2: Scan database for **exact matching** with the list of words complied in step 1.



Coincidencias (*word hits*) entre dos secuencias



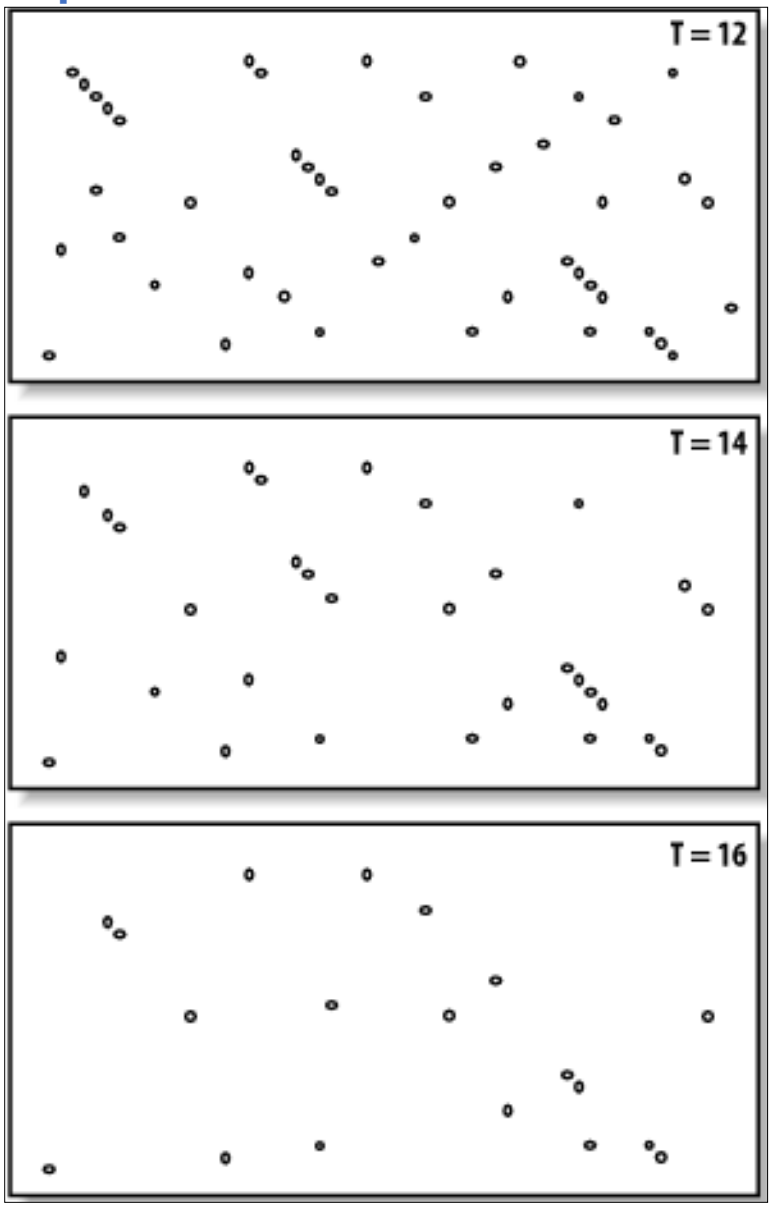
Cada punto en la matriz es un hit, es decir un match exacto entre las Words del query y las words de las secuencias de la DB.

Cada match tendrá un Score según la matriz

Pero no todas las words tienen vecinos

2►Alineamiento Local

Step 2: Scan database for **exact matching** with the list of words complied in step 1.



Un valor de W pequeño aumenta la sensibilidad pero disminuye la velocidad.

Un valor de T elevado disminuye la sensibilidad (se reduce le número de "*hits*" y se puede perder algún alineamiento significativo) pero aumenta la velocidad.

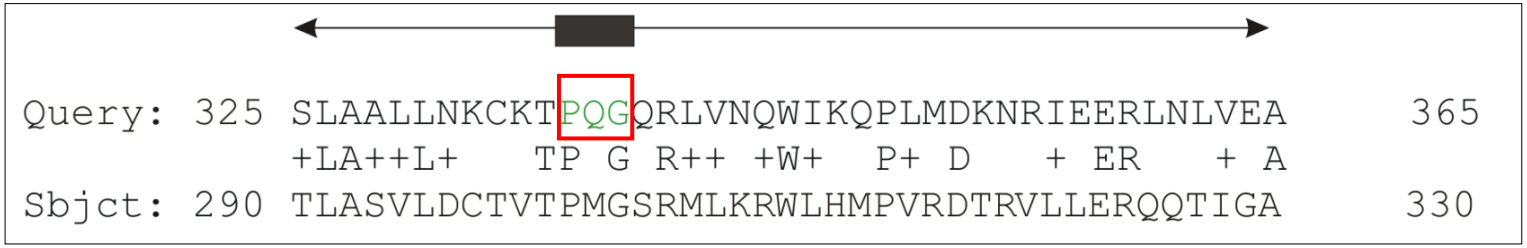
Una selección adecuada de W, T y la matriz de puntuación permite controlar de manera eficaz la sensibilidad y la rapidez del algoritmo

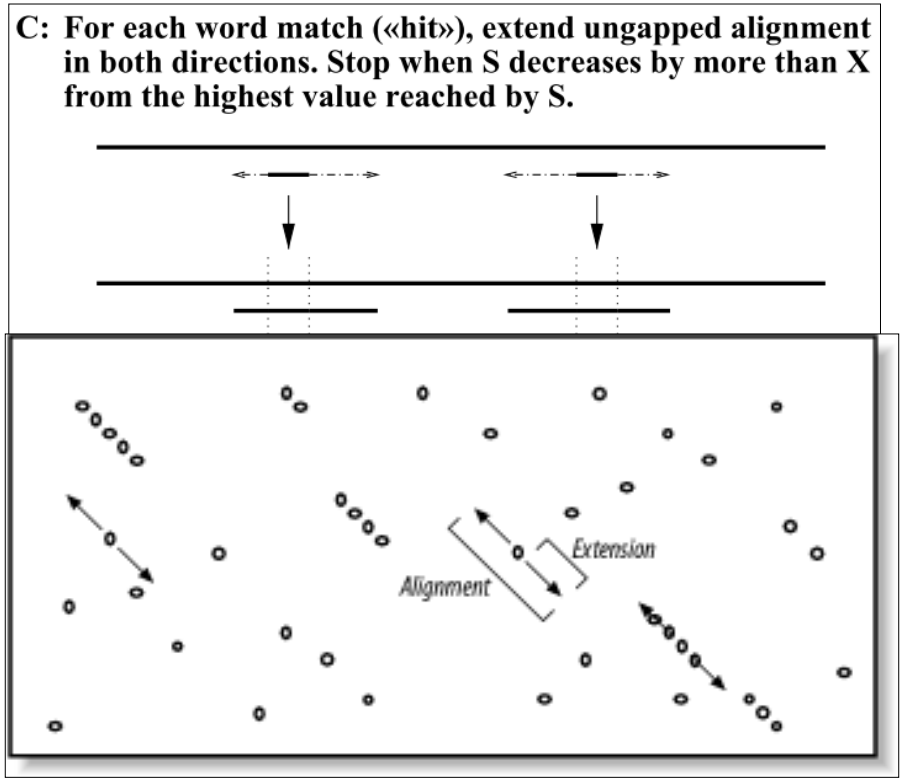
Efecto de los parámetros W (*word size*) y T (*threshold*)

3►Extension

Step 3: Extending hits from step 2.

BLAST1 intenta extender el alineamiento a ambos lados de cada coincidencia (sin dejar huecos), utilizando una variante del algoritmo de Smith-Waterman.





- Se extienden los hits de las words iniciales
- Las extensiones no agregan gaps a la alineación
- La secuencia se extiende a un HSP hasta que la puntuación de alineación cae por debajo de la puntuación máxima alcanzada menos un valor de parámetro de umbral
- Todas las HSP estadísticamente significativas son informadas

HSP: high scoring sequence pair – Es el score del alineamiento extendido hasta un score máximo

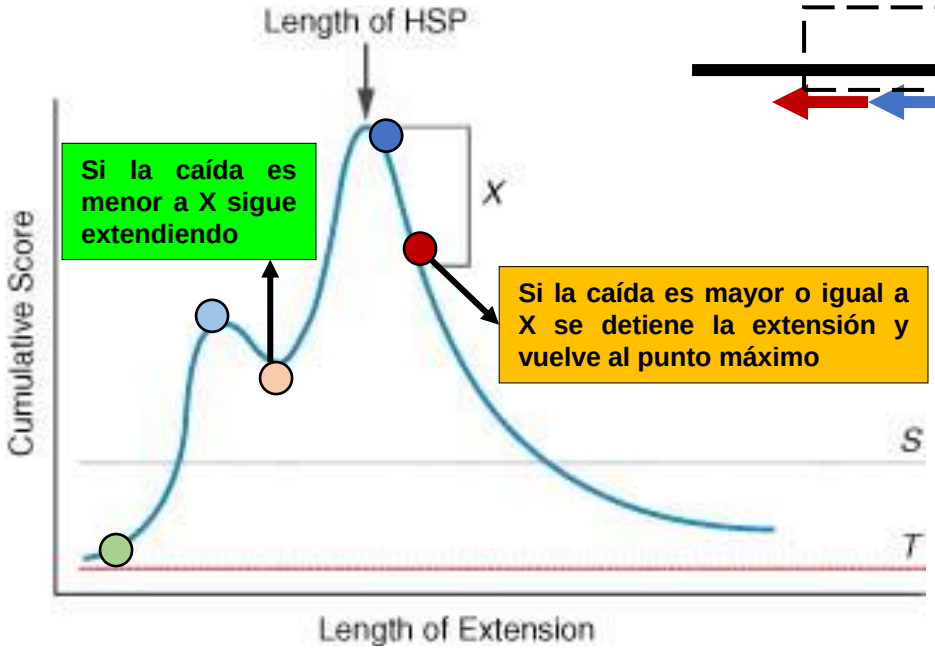
Tener vecinos cerca me asegura un mayor HSP

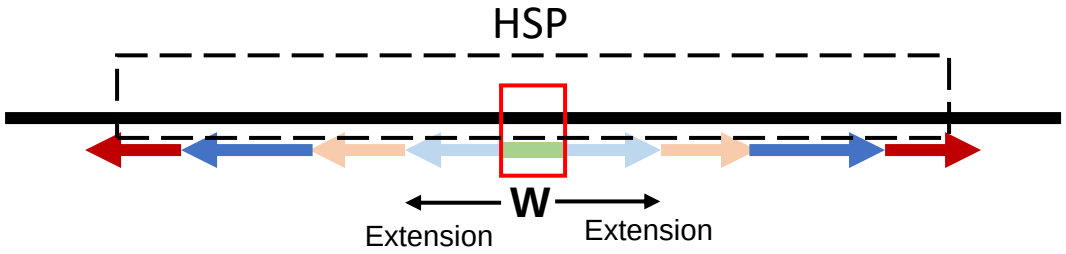
3►Extension

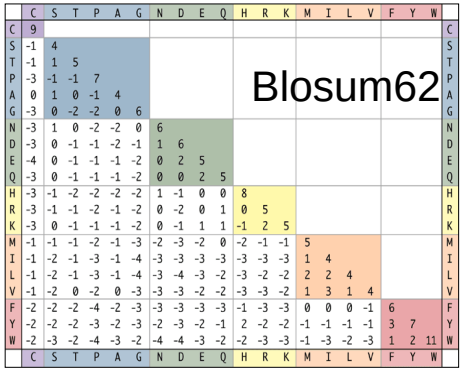
Step 3: Extending hits from step 2.

Once the initial word hit has been found, the **BLAST algorithm attempts to extend the match in the immediate sequence neighborhood.** Extension proceeds and the cumulative score is calculated using the scoring matrices discussed above. As long as positive matches and conservative substitutions outweigh the negative scores for gaps and mismatches**, the cumulative score will increase.** When the cumulative score starts to drops off, the BLAST algorithm measures the rate of decay and, **once past a certain point, stops trying to extend the alignment**. The result is an extended sequence alignment that was initially seeded by a word hit.

This is called an HSP, or high-scoring segment pair. All HSPs that have a cumulative scoreabove a certain threshold are reported in BLAST reports. Because the BLAST algorithm carries out these searches using all possible query words, it is possible that more than one HSP may be found for any given pair of sequences.







► Extender las palabras que han superado el umbral en ambas direcciones intentando mejorar la puntuacion

► La extensión concluye si la puntuacion desciende por debajo de otro umbral, si llega a cero, o si se acaba la secuencia.

Step 4: Evaluating significance of extended hits from step 3.

BLAST

W (tamaño inicial de palabra): W=11 (blastn)
W=28 (megablast)
W=3 (blastp)

S: valor del score total cuando termina la extensión. Para normaliar el valor del score y darle unidades (bits), se calcula el score normalizado:

$$(S') = \frac{\lambda S - \ln k}{\ln 2}$$

λ y K son unas constantes que dependen, respectivamente, del sistema de puntuación utilizado y del tamaño del espacio de secuencias m·n (siendo m el tamaño de la query y n el tamaño de la base de datos).

E-value: probabilidad de que una secuencia de tamaño similar dé el mismo *score* por azar en esta base de datos. $E = k \cdot m \cdot n \cdot e^{-\lambda S}$. O bien, si se utilizado el score normalizado, $E = m \cdot n \cdot 2^{-S'}$

Step 4: Evaluating significance of extended hits from step 3.

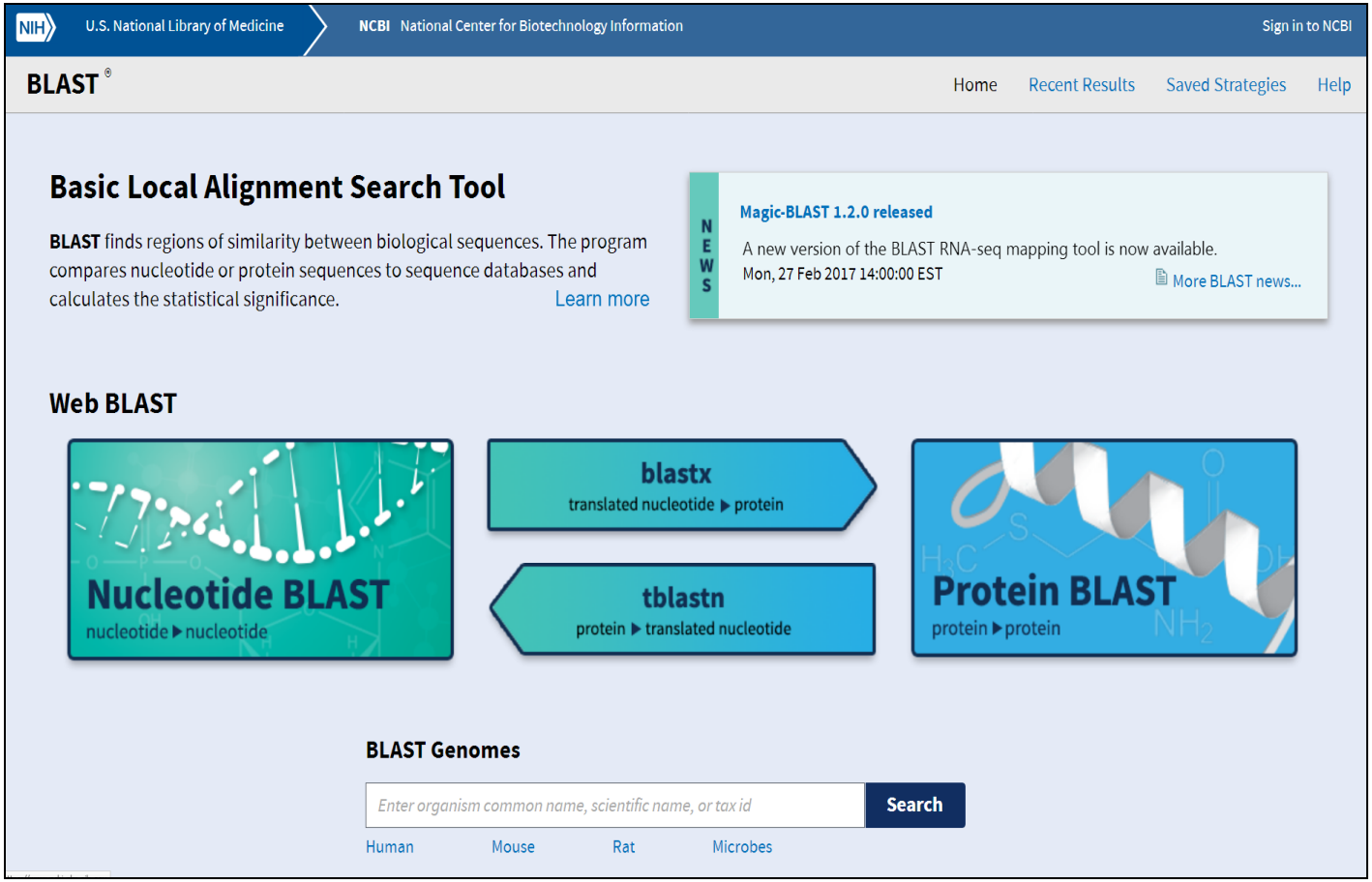
The **E-value (E)**, or expectation value is a good measure of the significance of the alignment. The E-value is the number of different alignments, with scores equivalent to or better than S, that are expected to occur in a database search by chance. The lower the E-value, the more significant the alignment result.

The E-value

The *expect value*(E-value) can be changed in order to limit the number of hits to the most significant ones. The lower the E-value, the better the hit. The E-value is dependent on the length of the query sequence and the size of the database. For example, an alignment obtaining an E-value of 0.05 means that there is a 5 in 100 chance of occurring by chance alone.

E-values are very dependent on the query sequence length and the database size. Short identical sequence may have a high E-value and may be regarded as "false positive" hits. This is often seen if one searches for short primer regions, small domain regions etc. The default threshold for the E-value on the BLAST web page is 10. Increasing this value will most likely generate more hits. Below are some rules of thumb which can be used as a guide but should be considered with common sense.

- **E-value < 10e-100** Identical sequences. You will get long alignments across the entire query and hit sequence.
- **10e-100< E-value < 10e-50** Almost identical sequences. A long stretch of the query protein is matched to the database.
- **10e-50 < E-value <10e-10** Closely related sequences, could be a domain match or similar.
- **10e-6 < E-value < 1** Could be a true homologue but it is a gray area.
- **E-value > 1** Proteins are most likely not related
- **E-value > 10** Hits are most likely junk unless the query sequence is very short.



There are many different BLAST programs available, but the ones most commonly used for basic database similarity searching are:

1. **blastp:** compares a protein sequence against a protein sequence database.
2. **blastn:** compares a nucleotide sequence against a nucleotide sequence database.
3. **blastx:** compares a six frame translation of a nucleotide sequence against a protein database
4. **tblastn:** compares a protein sequence against a six frame translation of a nucleotide database
5. **tblastx:** compares a six frame translation of a nucleotide sequence against a six frame translation of a nucleotide database.

BLAST2 (con GAPs)

Nucleic Acids Res. 25:3389-3402 (1997)

Nucleic Acids Research, 1997, Vol. 25, No. 17 3389–3402

Gapped BLAST and PSI-BLAST: a new generation of protein database search programs

Stephen F. Altschul*, Thomas L. Madden, Alejandro A. Schäffer[1], Jinghui Zhang, Zheng Zhang[2], Webb Miller[2] and David J. Lipman

National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894, USA, [1]Laboratory of Genetic Disease Research, National Human Genome Research Institute, National Institutes of Health, Bethesda, MD 20892, USA and [2]Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802, USA

Gapped BLAST and PSI-BLAST: a new generation of protein database search programs

Por: Altschul, SF (Altschul, SF); Madden, TL (Madden, TL); Schaffer, AA (Schaffer, AA); Zhang, JH (Zhang, JH); Zhang, Z (Zhang, Z); Miller, W (Miller, W); Lipman, DJ (Lipman, DJ)

Número de todas las veces citado

En Colección principal de Web of Science

47.705

49,074 en Todas las bases de datos

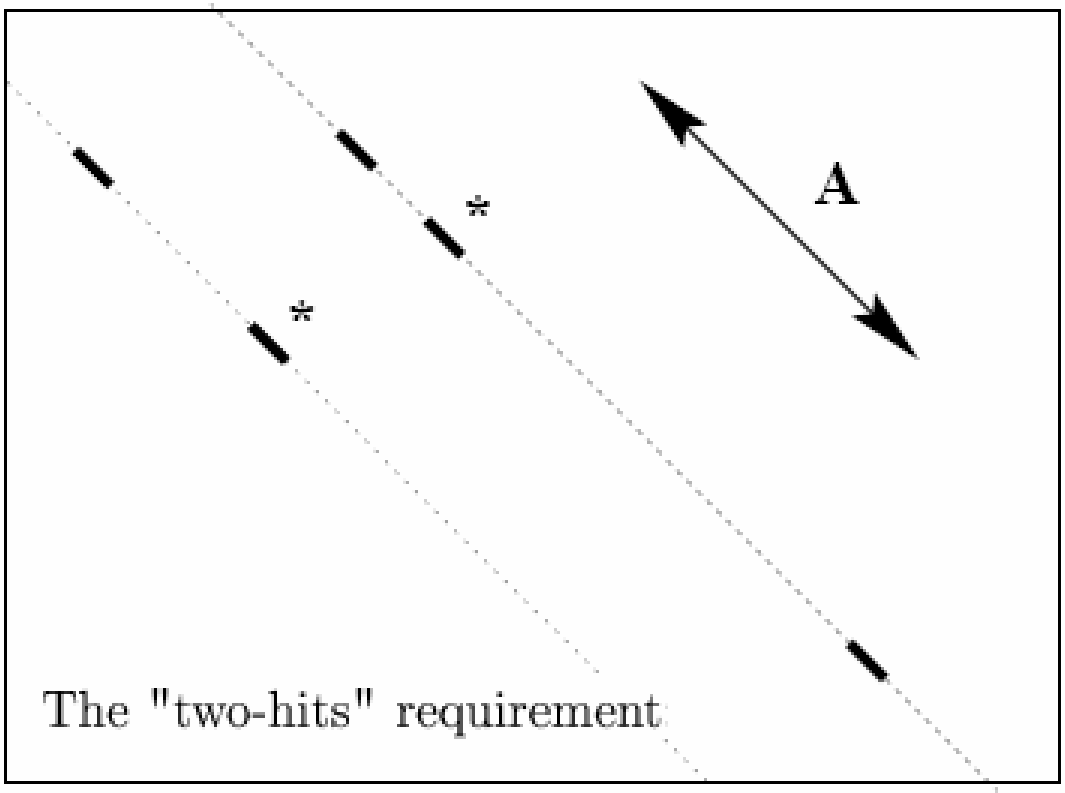
Etapa nº 3: algoritmo de la “doble coincidencia”

The new gapped BLAST algorithm (the two hit method)

(a) find two hits of score higher than T,

within a distance A.

(b) invoke an **ungapped** extension on the second hit.



those hits that are going to trigger an ungapped extension are labelled with a star

BLAST-2 utiliza el algoritmo de la doble coincidencia (*two-hit algorithm*): una palabra sólo se extiende (sin huecos) si existe otra en la misma diagonal a una distancia menor que A. El valor del parámetro A lo establece el usuario.

Esta extensión genera una serie de alineamientos con una puntuación elevada (HSP, *high scoring pairs*)

Etapa nº 3: algoritmo de la “doble coincidencia”

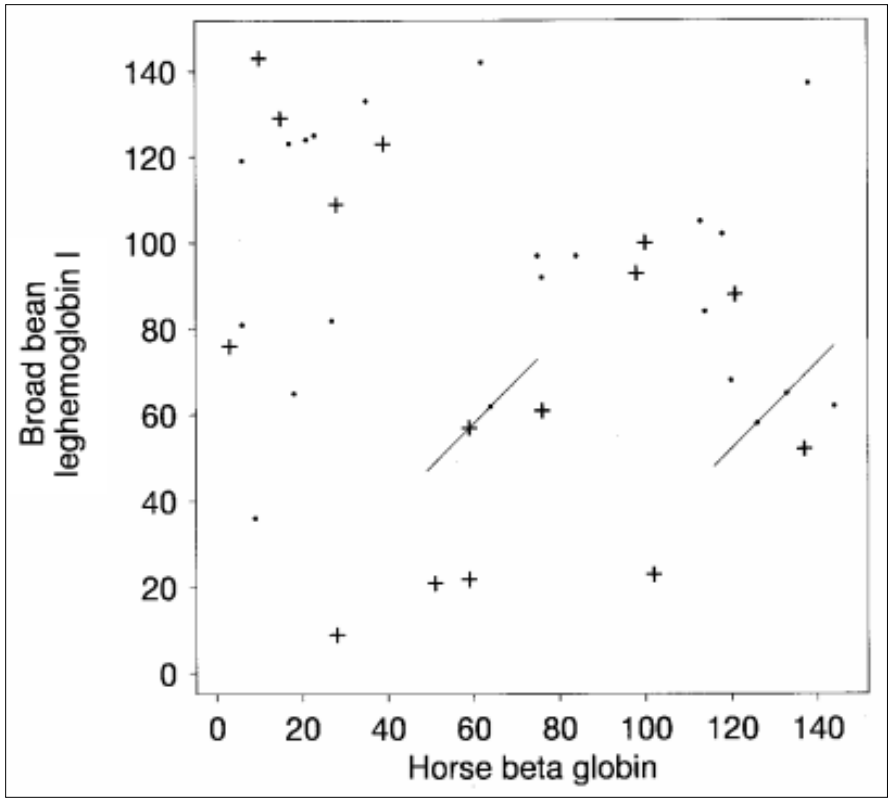
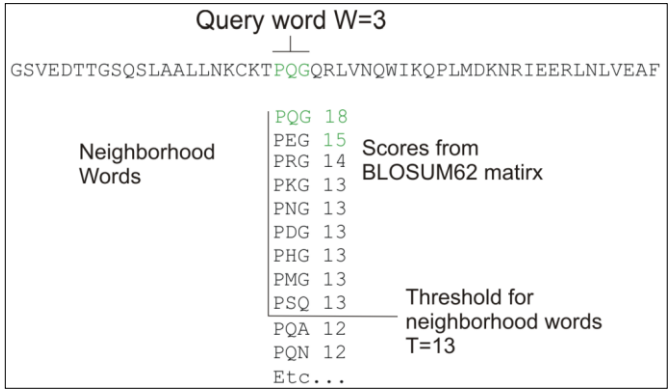


Figure 2. The BLAST comparison of broad bean leghemoglobin I (87) (SWISS-PROT accession no. P02232) and horse β-globin (88) (SWISS-PROT accession no. P02062). The 15 hits with score at least 13 are indicated by plus signs. An additional 22 non-overlapping hits with score at least 11 are indicated by dots. Of these 37 hits, only the two indicated pairs are on the same diagonal and within distance 40 of one another. Thus the two-hit heuristic with $T = 11$ triggers two extensions, in place of the 15 extensions invoked by the one-hit heuristic with $T = 13$.

Este requisito reduce la sensibilidad del método (se extienden menos palabras). Esta circunstancia se puede compensar disminuyendo el parámetro T (el umbral de puntuación que se utiliza en la primera etapa para generar la lista de “palabras vecinas”).

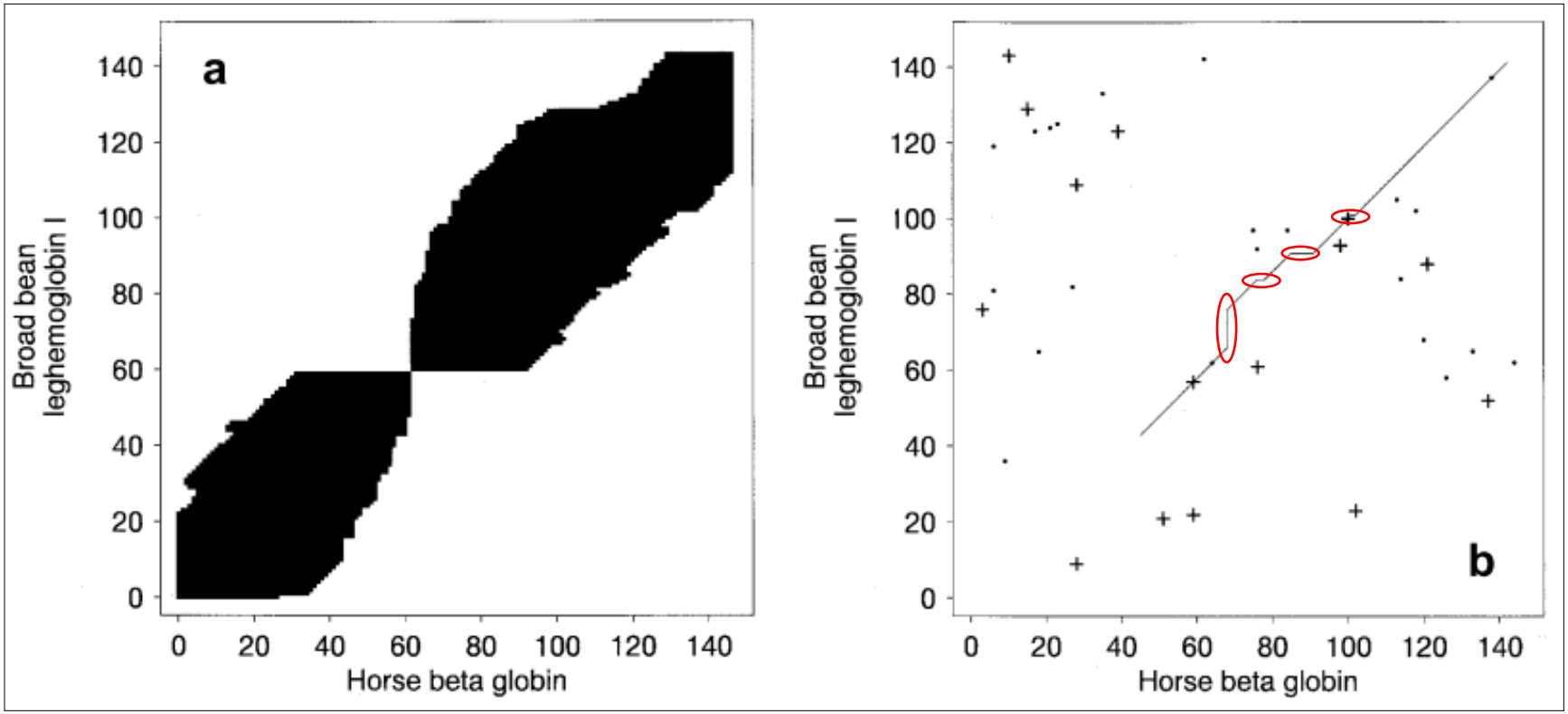


Se usa t=11 en lugar de T=13

Se reduce T para compensar la menor sensibilidad

Se hace una extensión con huecos en los mejores HSP

2. If the HSP generated has an expected score:
   (a) Trigger a **gapped** extension
   (b) If the final score has a significant E-value – report the gapped alignment.

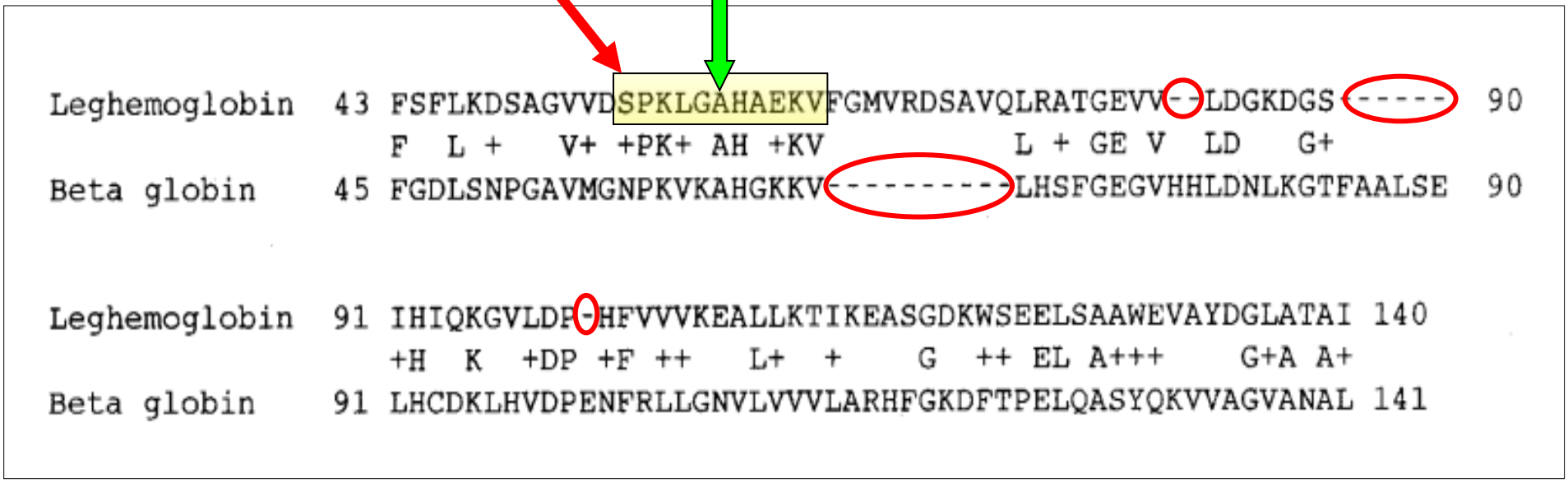


¿Dónde empieza el alineamiento con huecos?

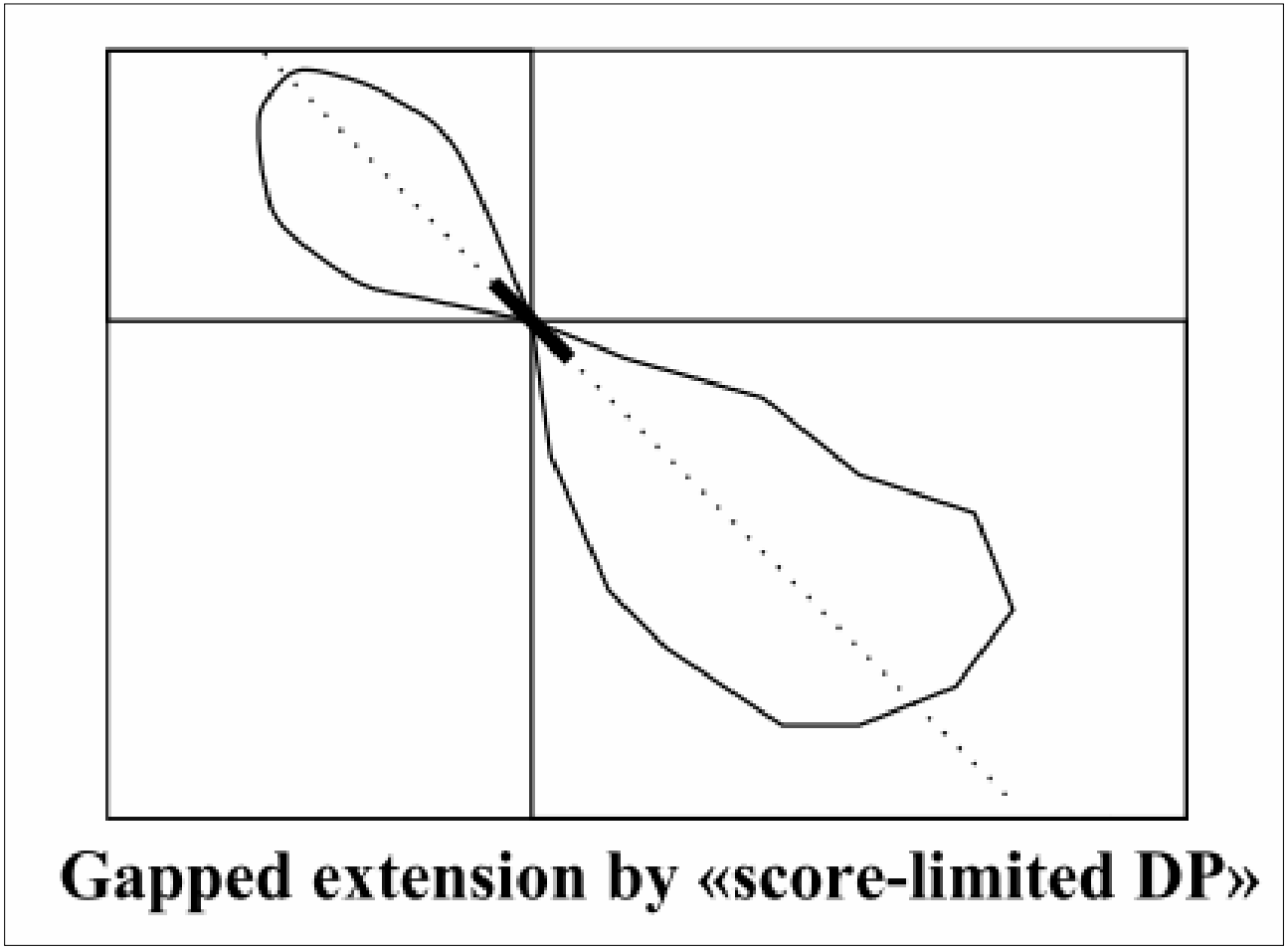
The algorithm used for computing these local gapped alignments is a modification of the Smith-Waterman algorithm: the Dynamic Programming matrix is explored in both directions (see figure 4D) starting from the middle point of the highest scoring $11 - tuple$ within the HSP, and the search for the optimal path is restricted to cells of the matrix such that the score of the alignment does not drop off more than X_g when compared to the maximal score reached since beginning the extension

Subsecuencia del HSP de 11 caracteres con la máxima puntuación

Residuo central de Alanina donde comienza, en ambas direcciones, el alineamiento local con huecos



El alineamiento local con huecos se lleva a cabo en ambas direcciones siempre y cuando la máxima puntuación alcanzada no se reduzca en un valor superior a X_g.



¿Dónde acaba el alineamiento con huecos?

BLAST

- BLAST is an algorithm for comparing primary biological sequence information like nucleotide or amino acid sequences
- Uses local sequence alignment
- Searches similarities in local alignment by comparing individual residues in the two sequences
- Better for similarity searching in closely matched or locally optimal sequences
- Works best for protein searches
- A sensitive bioinformatics tool
- Gaps between query and target sequences are not allowed
- More faster than FASTA

VS

FASTA

- FASTA is a DNA and protein sequence alignment software package hosted by European Bioinformatics Institute
- Uses local sequence alignment first, then extends to global alignment
- Searches similarities in local alignments by comparing sequence patterns or words
- Better for similarity searching in less similar sequences
- Works best for nucleotide searches
- More sensitive than BLAST
- Gaps are allowed
- Comparatively slower