Ejercicios2_algoritmos

Tania Gonzalo y Daniel Parra

WORKSHEET

Ejercicio 1 (4 puntos)

En este ejercicio probarás el algoritmo Needleman-Wunsch en una secuencia corta de partes de hemoglobina (código PDB 1AOW) y mioglobina 1 (código PDB 1AZI). Aquí alineará la secuencia HGSAQVKGHG con la secuencia KTEAEMKASEDLKKHGT.

Las dos secuencias están dispuestas en una matriz en la Tabla 1. Las secuencias comienzan en la esquina superior derecha, y las penalizaciones por desfase inicial se enumeran en cada posición inicial de desfase. La penalización por desfase se considera -8. Las puntuaciones de similitud Si,j procedentes de la búsqueda de coincidencias proceden de la tabla BLOSUM40.

Para resolver este ejercicio, hemos aplicado el algoritmo de Needleman-Wunsch para alinear dos secuencias de proteínas que nos indican: una parte de la hemoglobina (HGSAQVKGHG) y una parte de la mioglobina (KTEAEMKASEDLKKHGT). Este algoritmo de alineamiento global nos permite encontrar la mejor correspondencia entre las secuencias considerando inserciones, eliminaciones y sustituciones de aminoácidos.

Para llevar a cabo la alineación, seguimos los siguientes pasos:

1. **Inicialización de la matriz**: Se construyó una matriz en la que las secuencias se dispusieron en los ejes horizontal y vertical. Se asignaron penalizaciones de apertura de brecha (-8) en la primera fila y la primera columna.
2. **Cálculo de los valores en la matriz**: Cada celda se llenó utilizando la fórmula de Needleman-Wunsch:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(i,j) & \text{(match o mismatch según la matriz BLOSUM40)} \\ F(i-1,j) + \text{gap penalty} & \text{(inserción)} \\ F(i,j-1) + \text{gap penalty} & \text{(eliminación)} \end{cases}$$

Donde $S(i, j)$ es el puntaje de sustitución obtenido de la matriz BLOSUM40.

La tabla obtenida se muestra en Figura 1 donde aparecen todas las celdas rellenas además de las flechas indicativas de la procedencia de cada uno de los números para posteriormente poder llevar a cabo la recontrucción del alineamiento.

3. **Obtención del score final**: Una vez completada la matriz, el score final de alineación se encuentra en la casilla inferior derecha (Figura 1). Como podemos observar, en nuestro caso el score optenido es de -21.

4. **Retroceso para encontrar las alineaciones óptimas**: Siguiendo las flechas que indicaban la mejor elección en cada celda, realizamos un recorrido inverso para reconstruir los alineamientos óptimos. Encontramos dos posibles caminos (alternativas), lo que dio lugar a dos alineaciones diferentes, ambos mostrados en la parte inferior de la Figura 1.

Este procedimiento nos permitió obtener una alineación óptima de las secuencias, teniendo en cuenta tanto coincidencias como penalizaciones por inserciones y eliminaciones.

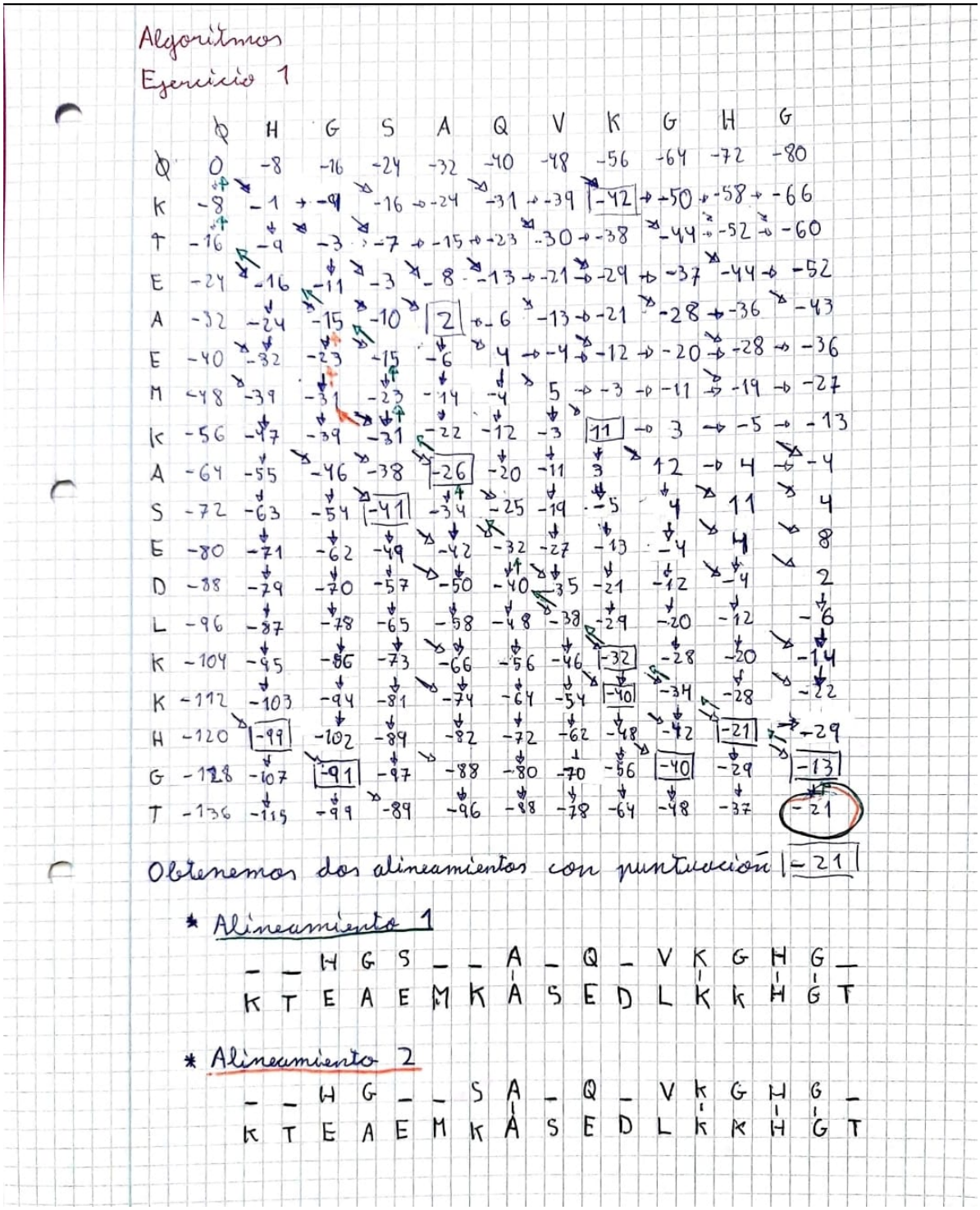


Figura 1: Solución manual del ejercicio 1. En la parte superior se mmuestra la matriz obtenida tras la aplicación del algoritmo de Needleman-Wunsch. En la parte inferior se muestran los dos alineamientos óptimos obtenidos.

Ejercicio 2 (6 puntos)

Dado el conjunto de secuencias múltiples:

- *S1: PPGVKSDCAS*
- *S2: PADGVKDCAS*
- *S3: PPDGKSDS*
- *S4: GADGKDCCS*
- *S5: GADGKDCAS*

Utilice el popular método de alineación progresiva para alinear globalmente el conjunto anterior de secuencias. Genere el árbol guía por unión de vecinos. Compare su resultado (alineamiento) con el de Clustal-Omega.

Con el alineamiento final representa el logo. Para este proposito los caracteres nulos o gap son ignorados y no cuentan para el número de observaciones de una columna.

Para la resolución de este ejercicio nos piden levar a cabo un alineación global de secuencias mediante alineación progresiva, donde el conjunto de secuencias con las que se trabaja es:

Secuencia	Cadena
S1	PPGVKSDCAS
S2	PADGVKDCAS
S3	PPDGKSDS
S4	GADGKDCCS
S5	GADGKDCAS

Metodología

Para este ejercicio se utilizó el método de alineación progresiva para realizar el alineamiento global de las secuencias. Se llevaron a cabo los siguientes pasos:

- **Alineamientos por pares**: Se utilizaron las herramientas EMBOSS Needle y Pairwise Sequence Alignment (PSA) para generar alineamientos dos a dos entre las secuencias.
- **Construcción de la matriz de distancias**: A partir de los alineamientos, se calculó una matriz de distancias basada en los valores de similitud entre las secuencias de forma manual.
- **Generación del árbol guía de forma manual**: Se empleó el método de unión de vecinos (Neighbor Joining) para construir el árbol guía a partir de la matriz de distancias.

- **Alineamiento progresivo manual**: Utilizando el árbol guía, se realizó el alineamiento progresivo de las secuencias.
- **Comparación con Clustal Omega**: Se comparó el alineamiento obtenido con el generado por Clustal Omega (1.2.4).
- **Generación del logo de secuencias**: Finalmente, se representó el alineamiento en formato de logo ignorando los caracteres nulos o gaps mediante el uso de las siguientes librerías en Python para el análisis y visualización de secuencias biológicas: `Biopython` (módulo AlignIO) para la manipulación de alineamientos de secuencias, `Logomaker` para la generación de logotipos de secuencia, `Matplotlib` para la visualización gráfica y `NumPy` para el procesamiento numérico.

1. Alineamientos por pares

Para los alineamientos por pares se utilizaron los siguientes parámetros:

- **Matriz de sustitución**: BLOSUM40
- **Penalización por gap**: -10 (aproximado a -8, usado en el ejercicio anterior)

Estos también se muestran en la figura Figura 2.

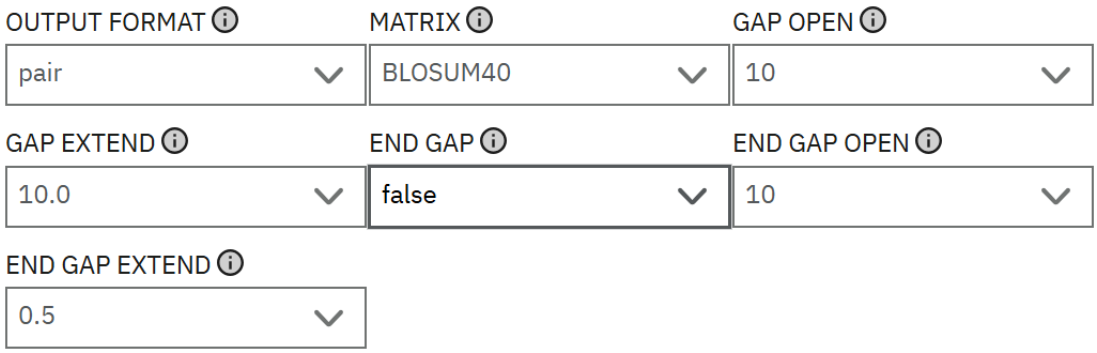


Figura 2: Parámetros para el alineamiento dos a dos.

Se calcularon los alineamientos y sus respectivos scores, definidos como la relación entre el número de coincidencias y la cantidad total de residuos (excluyendo gaps). Los resultados obtenidos fueron los siguientes:

Secuencias comparadas	Longitud 1	Longitud 2	Coincidencias	Score
S1 vs S2	10	10	8	8/10
S1 vs S3	10	8	5	5/9
S1 vs S4	10	9	4	8/19

Secuencias comparadas	Longitud 1	Longitud 2	Coincidencias	Score
S1 vs S5	10	9	5	10/19
S2 vs S3	10	8	4	4/9
S2 vs S4	10	9	7	14/19
S2 vs S5	10	9	8	16/19
S3 vs S4	8	9	4	8/17
S3 vs S5	8	9	3	6/17
S4 vs S5	9	9	8	8/9

Los resultados de los alineamientos obtenidos los podemos observar a continuación:

```
S1          1 P-PGVKSDCAS  10
              | .||| ||||
S2          1 PADGVK-DCAS  10
```

Figura 3: Alineamiento dos a dos de S1 vs S2, con 10 aa cada secuencia y 8 coincidencias, tiene un score 8/10.

```
S1          1 PPGVKSDCAS  10
              ||..|||.
S3          1 PPDGKSDS--   8
```

Figura 4: Alineamiento dos a dos de S1 vs S3, con 10 aa en S1 y 8 aa en S3; y 5 coincidencias, tiene un score 5/9.

```
S1          1 PPGVK-SDCAS  10
                |.. .||.|
S4          1 --GADGKDCCS   9
```

Figura 5: Alineamiento dos a dos de S1 vs S4, con 10 aa en S1 y 9 aa en S4; y 4 coincidencias, tiene un score 8/19.

```
S1          1 PPGVK-SDCAS    10
                |.. .||||
S5          1 --GADGKDCAS     9
```

Figura 6: Alineamiento dos a dos de S1 vs S5, con 10 aa en S1 y 9 aa en S5; y 5 coincidencias, tiene un score 10/19.

```
S2          1 PADGVKDCAS     10
              |.||..|.
S3          1 PPDGKSDS--      8
```

Figura 7: Alineamiento dos a dos de S2 vs S3, con 10 aa en S2 y 8 aa en S3; y 4 coincidencias, tiene un score 4/9.

```
S2          1 PADGVKDCAS     10
              .||| |||.|
S4          1 GADG-KDCCS      9
```

Figura 8: Alineamiento dos a dos de S2 vs S4, con 10 aa en S2 y 9 aa en S4; y 7 coincidencias, tiene un score 14/19.

```
S2          1 PADGVKDCAS     10
              .||| |||||
S5          1 GADG-KDCAS      9
```

Figura 9: Alineamiento dos a dos de S2 vs S5, con 10 aa en S2 y 9 aa en S5; y 8 coincidencias, tiene un score 16/19.

```
S3          1 PPDGKSDS--      8
              ..||| |.
S4          1 GADGK-DCCS      9
```

Figura 10: Alineamiento dos a dos de S3 vs S4, con 8 aa en S3 y 9 aa en S4; y 4 coincidencias, tiene un score 8/17.

```
S3                  1 PPDGKSDS-      8
                      ..|||..:
S5                  1 GADGKDCAS      9
```

Figura 11: Alineamiento dos a dos de S3 vs S5, con 8 aa en S3 y 9 aa en S5; y 3 coincidencias, tiene un score 6/17.

```
S4                  1 GADGKDCCS      9
                      |||||||.|
S5                  1 GADGKDCAS      9
```

Figura 12: Alineamiento dos a dos de S4 vs S5, con 9 aa cada secuencia y 8 coincidencias, tiene un score 8/9.

2. Construcción de la matriz de distancias

La matriz de distancias se construyó a partir del score de similitud, aplicando la transformación:

$$d = 1 - \text{score}$$

Obteniendo la siguiente matriz de distancias que se puede observar en la parte superior de Figura 13:

	S1	S2	S3	S4	S5
S1	0.00	0.20	0.44	0.58	0.47
S2	0.20	0.00	0.56	0.26	0.16
S3	0.44	0.56	0.00	0.53	0.65
S4	0.58	0.26	0.53	0.00	0.11
S5	0.47	0.16	0.65	0.11	0.00

3. Construcción del árbol guía de forma manual

A partir de la matriz de distancias, se generó el arbol guía utilizando el método de unión de vecinos (*Neighbor Joining*). El árbol guía obtenido muestra la relación evolutiva entre las secuencias y nos sirve como base para la alineación progresiva. El proceso de generación del arbol guía se hizo de forma manueal y se muestra en Figura 13.

El arbol guía obtenido finalmente fue el mostrado en Figura 14.

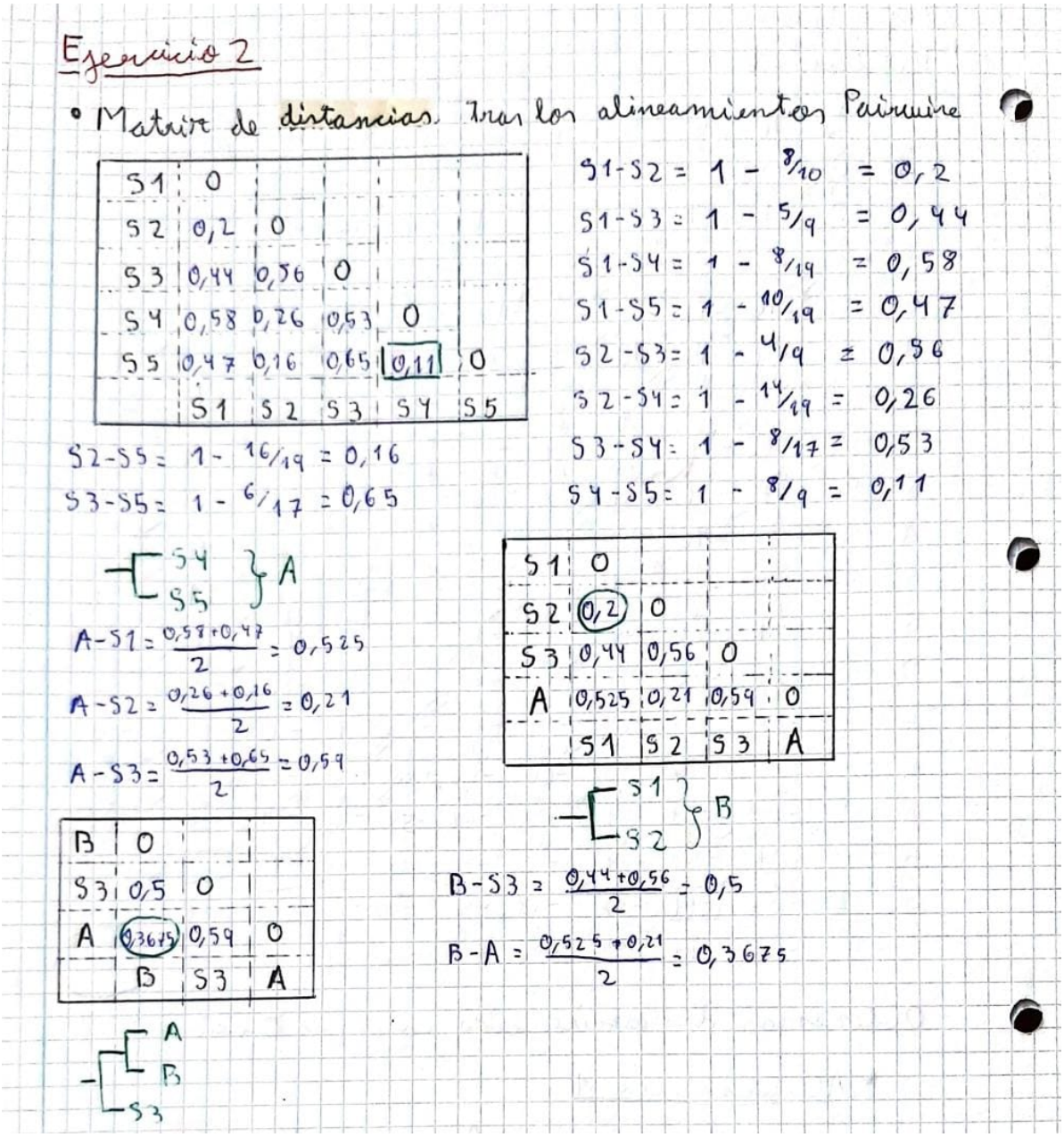


Figura 13: Construcción de matrices de distancias.

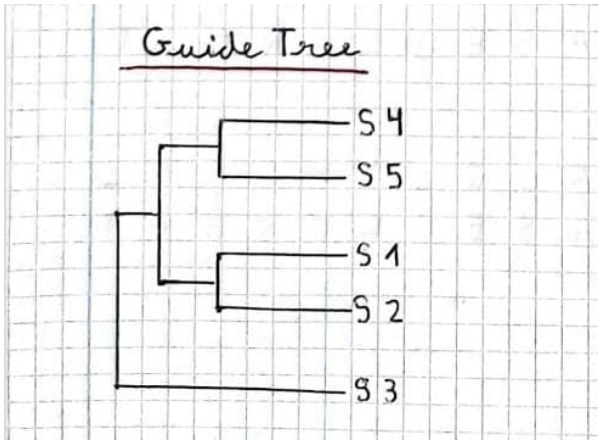


Figura 14: Guide Tree obtenido a partir de alineamientos 2 a 2 con parámetros similares al ejercicio 1.

4. Alineamiento progresivo manual

Utilizando el árbol guía, se llevó a cabo el alineamiento progresivo de las secuencias. Este alineamiento se obtuvo manualmente y posteriormente se comparará con el resultado de Clustal Omega.

El alineamiento final obtenido se muestra en Figura 15.

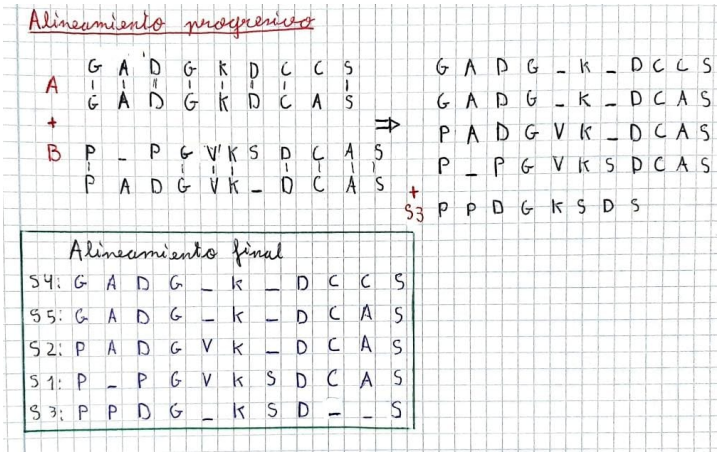


Figura 15: Alineamiento progresivo obtenido a partir de alineamientos 2 a 2 con parámetros similares al ejercicio 1.

Cálculo con parámetros por defecto

Además del cálculo utilizando BLOSUM40, se decidió repetir el proceso usando la matriz BLOSUM62 y los parámetros por defecto, para evaluar si el alineamiento final variaba en función de los parámetros. Los parámetros utilizados fueron los que se muestran en Figura 16.

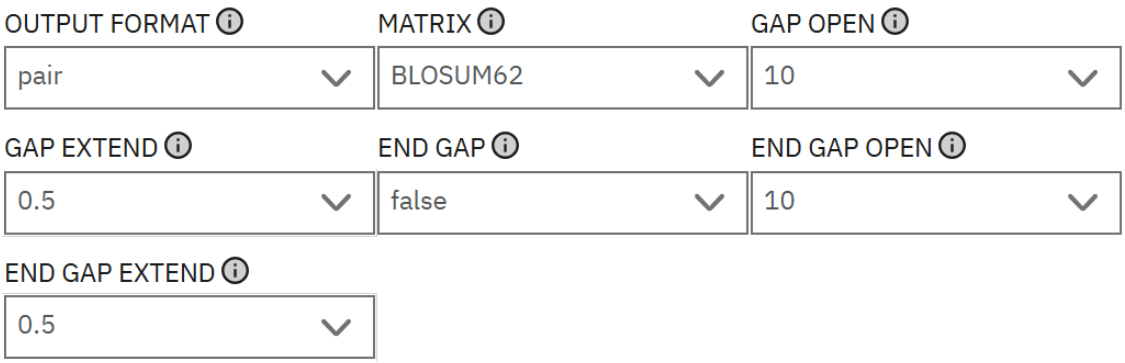


Figura 16: Parámetros para el alineamiento dos a dos por defecto.

Se realizaron los alineamientos por pares con estos parámetros y se generó una nueva matriz de distancias, un nuevo árbol guía y el alineamiento progresivo correspondiente.

Los scores obtenidos con estos parámetros fueron se muestran en la siguiente tabla:

Secuencias comparadas	Longitud 1	Longitud 2	Coincidencias	Score
S1 vs S2	10	10	7	7/10
S1 vs S3	10	8	5	5/9
S1 vs S4	10	9	4	8/19
S1 vs S5	10	9	5	10/19
S2 vs S3	10	8	4	4/9
S2 vs S4	10	9	7	14/19
S2 vs S5	10	9	8	16/19
S3 vs S4	8	9	3	6/17
S3 vs S5	8	9	3	6/17
S4 vs S5	9	9	8	8/9

Y los alineamientos obtenidos se muestran a continuación:

```
S1                  1 -PPGVKSDCAS     10
                      ..||| ||||
S2                  1 PADGVK-DCAS     10
```

Figura 17: Alineamiento dos a dos con parámetros por defecto, de S1 vs S2, con 10 aa cada secuencia y 7 coincidencias, tiene un score 7/10.

```
S1                  1 PPGVKSDCAS      10
                      ||..|||.
S3                  1 PPDGKSDS--       8
```

Figura 18: Alineamiento dos a dos con parámetros por defecto, de S1 vs S3, con 10 aa en S1 y 8 aa en S3; y 5 coincidencias, tiene un score 5/9.

```
S1                  1 PPGVK-SDCAS     10
                        |.. .||.|
S4                  1 --GADGKDCCS      9
```

Figura 19: Alineamiento dos a dos con parámetros por defecto, de S1 vs S4, con 10 aa en S1 y 9 aa en S4; y 4 coincidencias, tiene un score 8/19.

```
S1                  1 PPGVK-SDCAS     10
                        |.. .||||
S5                  1 --GADGKDCAS      9
```

Figura 20: Alineamiento dos a dos con parámetros por defecto, de S1 vs S5, con 10 aa en S1 y 9 aa en S5; y 5 coincidencias, tiene un score 10/19.

```
S2                  1 PADGVKDCAS      10
                      |.||..|.
S3                  1 PPDGKSDS--       8
```

Figura 21: Alineamiento dos a dos con parámetros por defecto, de S2 vs S3, con 10 aa en S2 y 8 aa en S3; y 4 coincidencias, tiene un score 4/9.

```
S2                1 PADGVKDCAS     10
                    .||| |||.|
S4                1 GADG-KDCCS      9
```

Figura 22: Alineamiento dos a dos con parámetros por defecto, de S2 vs S4, con 10 aa en S2 y 9 aa en S4; y 7 coincidencias, tiene un score 14/19.

```
S2                1 PADGVKDCAS     10
                    .||| |||||
S5                1 GADG-KDCAS      9
```

Figura 23: Alineamiento dos a dos con parámetros por defecto, de S2 vs S5, con 10 aa en S2 y 9 aa en S5; y 8 coincidencias, tiene un score 16/19.

```
S3                1 PPDGKSDS-       8
                    ..|||...
S4                1 GADGKDCCS       9
```

Figura 24: Alineamiento dos a dos con parámetros por defecto, de S3 vs S4, con 8 aa en S3 y 9 aa en S4; y 3 coincidencias, tiene un score 6/17.

```
S3                1 PPDGKSDS-       8
                    ..|||..:
S5                1 GADGKDCAS       9
```

Figura 25: Alineamiento dos a dos con parámetros por defecto, de S3 vs S5, con 8 aa en S3 y 9 aa en S5; y 3 coincidencias, tiene un score 6/17.

```
S4                1 GADGKDCCS       9
                    |||||||.|
S5                1 GADGKDCAS       9
```

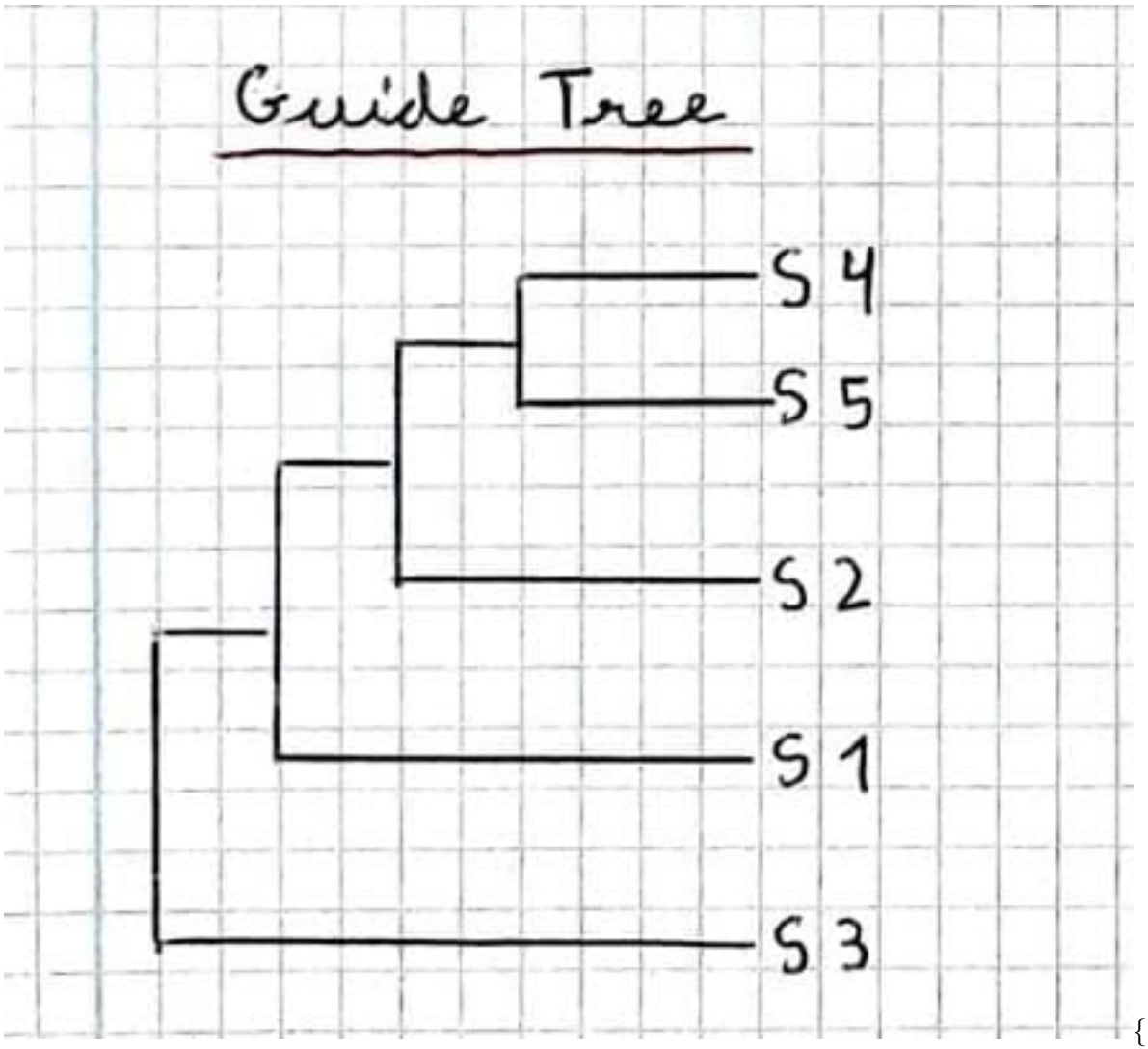
Figura 26: Alineamiento dos a dos con parámetros por defecto, de S4 vs S5, con 9 aa cada secuencia y 8 coincidencias, tiene un score 8/9.

Con ello, contruimos la matriz de distancias:

	S1	S2	S3	S4	S5
S1	0.00	0.30	0.44	0.58	0.47
S2	0.30	0.00	0.56	0.26	0.16
S3	0.44	0.56	0.00	0.53	0.65
S4	0.58	0.26	0.53	0.00	0.11
S5	0.47	0.16	0.65	0.11	0.00

Y se llevó a cabo el alineamiento progresivo de forma análoga al anterior (Figura 27)

Finalmente, construimos el guide tree:



{#fig-

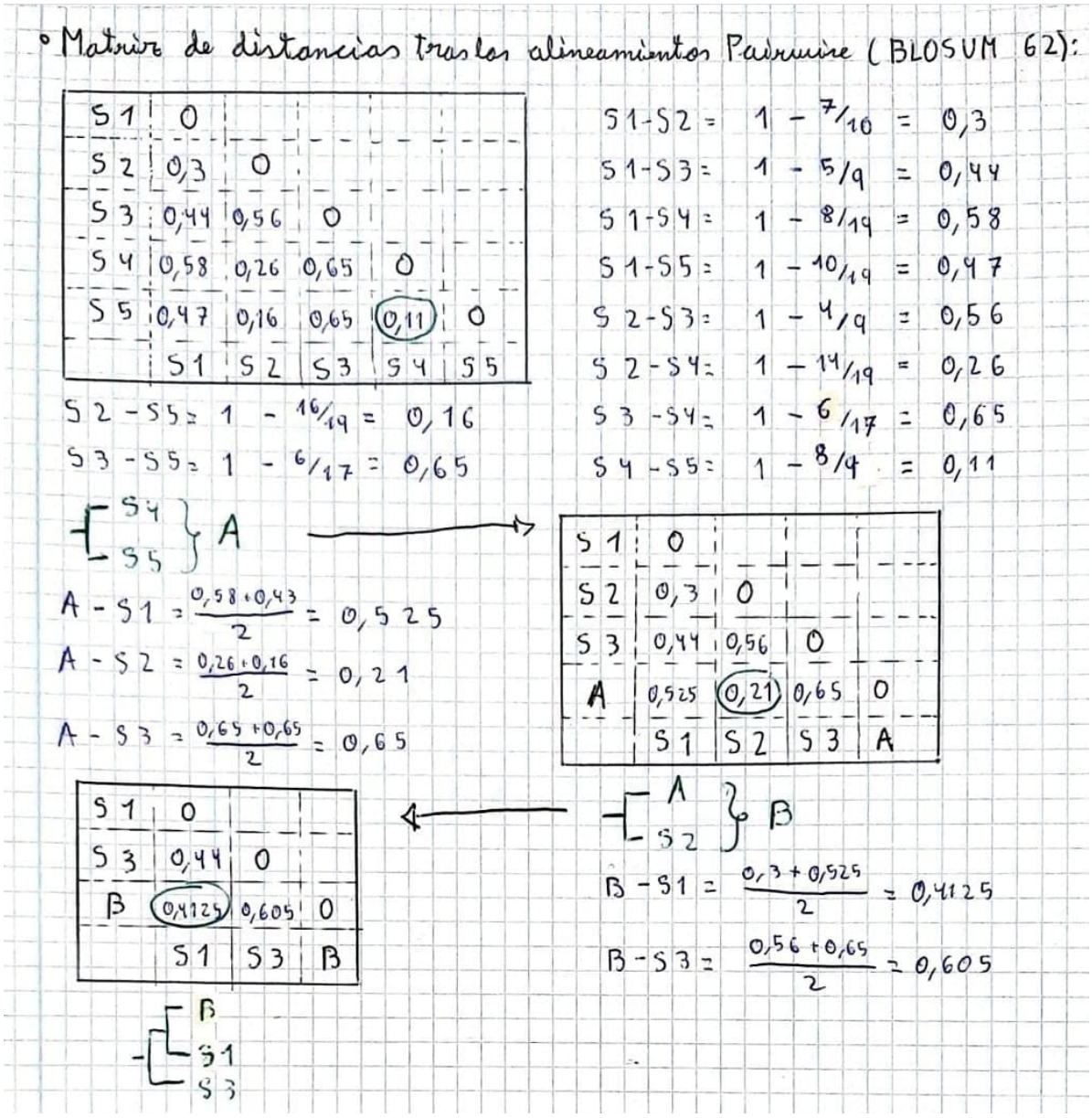


Figura 27: Construcción de matrices de distancias.

ej2-7, width="50%"}

Y el alineamiento progresivo se muestra en la figura Figura 28

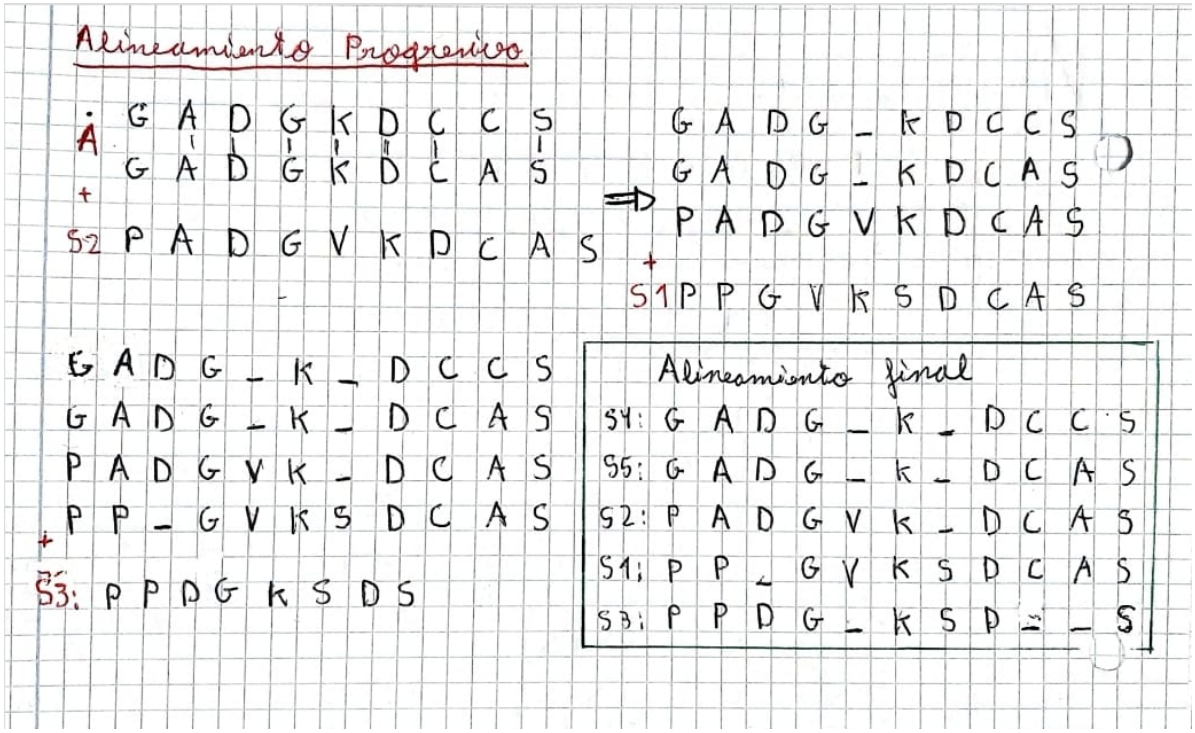


Figura 28: Alineamiento progresivo obtenido a partir de alineamientos 2 a 2 con matriz BLOSUM62

Alineamiento múltiple

Para llevar a cabo el alineamiento múltiple de secuencias, se utilizaron los siguientes parámetros:

Figura 29: Parámetros para el alineamiento múltiple.

Estos parámetros fueron configurados en Clustal Omega, una herramienta bioinformática para la alineación de secuencias.

Como resultado del proceso, se obtuvo el siguiente árbol guía, que representa la relación filogenética entre las secuencias alineadas:

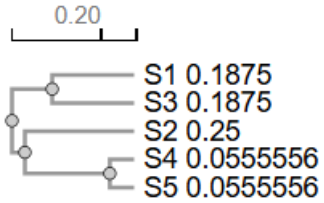


Figura 30: Guide tree del alineamiento múltiple por Clustal Omega.

Finalmente, se generó el alineamiento múltiple de las secuencias, donde se pueden observar las similitudes y diferencias entre ellas:

Este alineamiento nos permite analizar la conservación de regiones específicas, identificar mutaciones y establecer relaciones evolutivas entre las secuencias comparadas.

Resultados

Comparación alineamientos

1. Primer alineamiento (manual, con BLOSUM40)(Figura 15): Existe una brecha significativa en S3 (representada por guiones), lo que indica una menor similitud con las demás

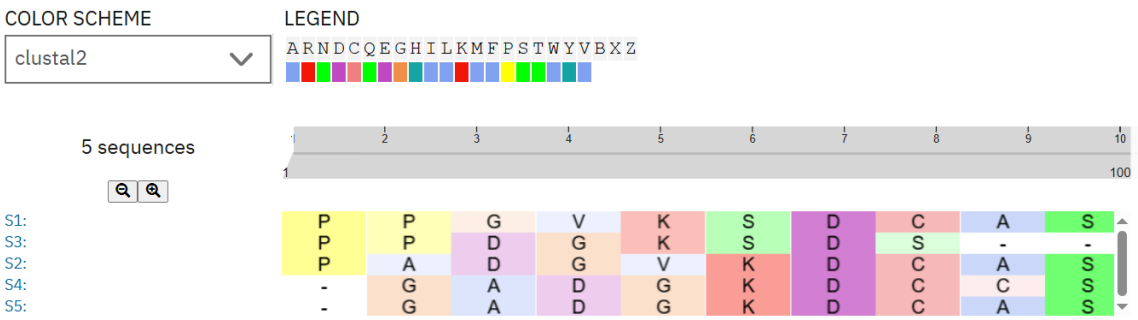


Figura 31: Alineamiento múltiple por Clustal Omega.

secuencias. Las secuencias S4 y S5 muestran una alta conservación en sus posiciones (similitud cercana en aminoácidos).

2. Segundo alineamiento (manual, con BLOSUM62) (Figura 28): Aparece una ligera variación en S1 (gap introducido en tras la segunda prolina (P) en lugar de tras la primera prolina (P). El resto de las secuencias mantiene el mismo patrón que en el primer alineamiento.

Esto es debido a la utilización de distintas matrices de sustitución donde la matriz BLOSUM62 favorece la detección de similitudes más estrechas en secuencias altamente relacionadas y también cómo la matriz penaliza los gaps o prioriza substituciones específicas.

3. Tercer alineamiento (Clustal Omega)(Figura 31): En este caso podemos ver que el alineamiento difiere bastante a los obtenindos de forma manual. Podemos observar menos gaps y, en caso de haberlos, estos se encuentran en los extremos (menor número de brechas y una mayor continuidad), es decir, presenta un mayor número de aminoácidos alineados, lo que puede ser el resultado de la automatización del proceso.

Con todo ello, podemos concluir que en todos los casos, S4 y S5 presentan una alta conservación, consolidando que están estrechamente relacionadas. La mayor diferencia se encuentra en la cadena S3, donde los métodos manuales marcan a S3 como más divergente (más gaps), pero Clustal Omega lo integra más en el alineamiento (mismatch en lugar de gaps).

Esto puede deberse a que Clustal Omega se basa en un algoritmo heurístico y, por tanto, no garantiza encontrar el alineamiento globalmente óptimo, teniendo dificultad frente a deleciones e inserciones, en las que incluye gaps. Además, debemos tener en cuenta la importancia de los parámetros introducidos para que esta se ajuste a la realidad. En el peor de los casos se utilizan valores predeterminados (como ocurre con Clustal Omega (1.2.4)), que pueden no ser ideales para todo el conjunto de datos. También hay que considerar que Clustal Omega usa el método HHalign, que es un algoritmo basado en modelos de Markov ocultos (HMM), para realizar alineamientos más precisos entre perfiles de secuencias.

Comparación de árboles guía

Como hemos mostrado anteriormente, se realizaron finalmente tres arboles diferentes (Figura 14, **?@fig-ej2-7**, Figura 30):

1. Árbol basado en BLOSUM40 (manual, alineamiento progresivo y vecinos cercanos):

- S4 y S5 son los más cercanos, agrupándose como un clado separado.
- S1 y S2 son también los más cercanos, agrupándose como un clado separado.
- Estos dos clado formados son unen en un antecesor común.
- S3 es el más distante, uniéndose finalmente al resto.

2. Árbol basado en BLOSUM62 (manual, alineamiento progresivo y vecinos cercanos):

La estructura difiere de la obtenida con la matriz BLOSUM40. En este caso: - S4 y S5 están más estrechamente relacionados. - S2 se incorpora al grupo S4-S5. - S1 se une después al conjunto principal. - S3 es el más distante, uniéndose finalmente al resto.

Esto refleja que cambiar la matriz de sustitución (BLOSUM40 vs. BLOSUM62) afecta a las relaciones globales en la construcción del arbol guía mediante este método.

3. Árbol generado por Clustal Omega: Los resultados en este caso se presentan en un formato numérico con distancias específicas:

- S4 y S5 siguen siendo los más cercanos, con una distancia de 0.055556 .
- S2 se sitúa más lejos de S4 y S5, con una distancia de 0.25 .
- S1 y S3 forman un subgrupo con una distancia de 0.1875 , diferente a los árboles manuales.

Podemos observar que este método introduce diferencias en cómo se agrupan S1 y S3 en comparación con los anteriores.

Con todo ello, podemos concluir una relación S4-S5 consistente ya que en todos los métodos y matrices utilizadas, S4 y S5 son siempre los más cercanos, lo que indica una fuerte similitud entre estas secuencias. Además, hemos observado el impacto de qu tiene escoger una u otra matriz BLOSUM al comparar los resultados entre BLOSUM40 y BLOSUM62 y observando que el arbol guia obtenido es diferente. Por último, ninguno de los árboles obtenidos manuealmente coincide con el obtenido por el algoritmo Clustal Omega, que introduce distancias precisas e incluye la relación entre S1 y S3, no obtenido manualmente.

Generación del logo a patir de los alineamientos obtenidos

Para representar el alineamiento final en formato de logo, se utilizó el lenguaje de programación Python y las librerías `Biopython`, `logosmaker`, `matplotlib` y `numpy`.

En la construcción del logo, se ignoraron los gaps y se consideraron únicamente los residuos presentes en cada posición del alineamiento. Por ello, el número de aminoácido presentes es de 8 y, por tanto, el máximo valor que se puede obtener de cantidad de información (bits, representados en el eje y) es de 3.

1. Logo del alineamiento obtenido usando BLOSUM40

Cargamos las librerías necesarias, cargamos los documentos FASTA de los alineamientos progresivos y obtuvimos el logo correspondiente.

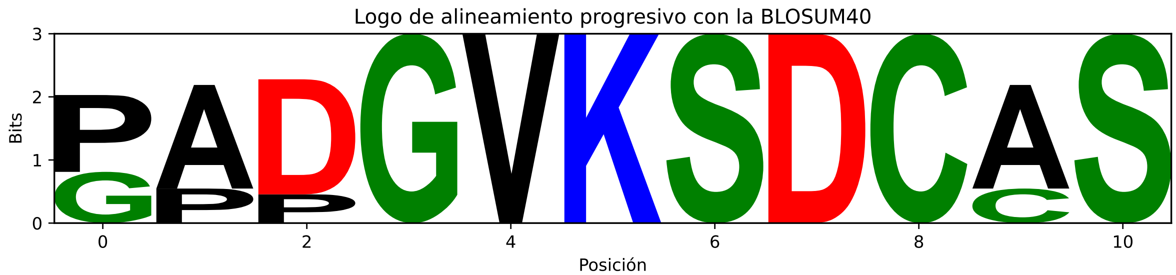
```
#Importación de librerías
from Bio import AlignIO
import logomaker
import matplotlib.pyplot as plt
import numpy as np
```

```
# Cargar y procesar el alineamiento
alignment = AlignIO.read("Alineamiento_final_BLOUSUM40.fa", "fasta")
sequences = [str(record.seq) for record in alignment]

# Crear matriz de conteos (sin gaps)
counts_matrix = logomaker.alignment_to_matrix(sequences,\
    to_type='counts',\
    characters_to_ignore='-') # Excluir gaps para evitar sesgos

# Normalizar a frecuencias relativas y convertir a información (bits)
information_matrix = logomaker.transform_matrix(counts_matrix, \
  from_type='probability', to_type='information')

# Generar el logo
logo = logomaker.Logo(
    information_matrix,
    color_scheme='chemistry', #Colores por propiedades químicas
    font_name='Arial',
    show_spines=True,
    stack_order='big_on_top' #Aminoácidos más frecuentes arriba
)

# Ajustar el estilo del gráfico
```

```
logo.ax.set_title("Logo de alineamiento progresivo con la BLOSUM40")
logo.ax.set_xlabel("Posición")
logo.ax.set_ylabel("Bits")
logo.ax.set_ylim(0, 3)

plt.tight_layout()
plt.show()
```

```
in validate_matrix(): Row sums in df are not close to 1. Reormalizing rows...
```

```
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  bg_df.loc[:, :] = 1 / num_cols
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  bg_df.loc[:, :] = 1 / num_cols
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  bg_df.loc[:, :] = 1 / num_cols
```



Podemos observar la representación gráfica del logo de alineamiento progresivo basado en la matriz de sustitución BLOSUM40. Este tipo de visualización permite evaluar la conservación y variabilidad de secuencias de aminoácidos en alineaciones múltiples, proporcionando información relevante sobre la evolución y estructura de las proteínas.

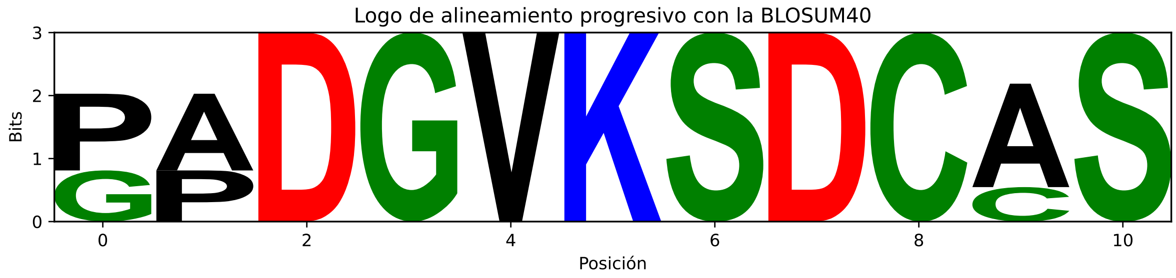
En el eje vertical aparecen representados los bits que representan la cantidad de información en cada posición de la secuencia. Valores altos indican una mayor conservación de los aminoácidos en esa posición. En el eje horizontal tenemos las posiciones de la secuencia donde podemos obervar la ubicación específica de los aminoácidos en la alineación. En el logo en sí aparecen representadas las letras correspondientes a un aminoácido concreto, y su tamaño refleja la frecuencia con la que aparece en una posición determinada.

Cada color indica propiedades de los aminoácidos de forma que los aminoácidos azules son los básicos; los rojos los aminoácidos ácidos; los verdes polares no cargados; los morados, aromáticos; los verde oscuro, residuos especiales y los negro, hidrofóbicos.

2. Logo del alineamiento obtenido usando BLOSUM62

```
# Cargar y procesar el alineamiento
alignment = AlignIO.read("Alineamiento_final_BLOUSUM62.fa", "fasta")
sequences = [str(record.seq) for record in alignment]

# Crear matriz de conteos (sin gaps)
counts_matrix = logomaker.alignment_to_matrix(sequences,\
    to_type='counts',\
    characters_to_ignore='-') # Excluir gaps para evitar sesgos

# Normalizar a frecuencias relativas y convertir a información (bits)
information_matrix = logomaker.transform_matrix(counts_matrix, \
  from_type='probability', to_type='information')

# Generar el logo
logo = logomaker.Logo(
    information_matrix,
    color_scheme='chemistry', #Colores por propiedades químicas
    font_name='Arial',
    show_spines=True,
    stack_order='big_on_top' #Aminoácidos más frecuentes arriba
)

# Ajustar el estilo del gráfico
logo.ax.set_title("Logo de alineamiento progresivo con la BLOSUM40")
logo.ax.set_xlabel("Posición")
logo.ax.set_ylabel("Bits")
logo.ax.set_ylim(0, 3)

plt.tight_layout()
plt.show()
```

```
in validate_matrix(): Row sums in df are not close to 1. Reormalizing rows...

C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\src
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\src
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\src
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\src
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\src
  bg_df.loc[:, :] = 1 / num_cols
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\src
  bg_df.loc[:, :] = 1 / num_cols
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\src
  bg_df.loc[:, :] = 1 / num_cols
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\src
  bg_df.loc[:, :] = 1 / num_cols
```



3. Logo del alineamiento obtenido usando Clustal Omega

```
# Cargar y procesar el alineamiento
alignment = AlignIO.read("clustalo-I20250321-165928-0743-92365021-p1m.fa", \
  "fasta")
sequences = [str(record.seq) for record in alignment]

# Crear matriz de conteos (sin gaps)
counts_matrix = logomaker.alignment_to_matrix(sequences,\
    to_type='counts',\
    characters_to_ignore='-') # Excluir gaps para evitar sesgos

# Normalizar a frecuencias relativas y convertir a información (bits)
information_matrix = logomaker.transform_matrix(counts_matrix, \
```

```
  from_type='probability', to_type='information')

# Generar el logo
logo = logomaker.Logo(
    information_matrix,
    color_scheme='chemistry', #Colores por propiedades químicas
    font_name='Arial',
    show_spines=True,
    stack_order='big_on_top' #Aminoácidos más frecuentes arriba
)

# Ajustar el estilo del gráfico
logo.ax.set_title("Logo de alineamiento progresivo con la BLOSUM40")
logo.ax.set_xlabel("Posición")
logo.ax.set_ylabel("Bits")
logo.ax.set_ylim(0, 3)

plt.tight_layout()
plt.show()
```

```
in validate_matrix(): Row sums in df are not close to 1. Reormalizing rows...
```

```
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
C:\Users\Daniel Parra\AppData\Local\Programs\Python\Python312\Lib\site-packages\logomaker\sr
  df.loc[:, :] = df.values / sums[:, np.newaxis]
```

De los tres logos obtenidos, podemos destacar que aquel con más variabilidad en las posiciones es el obtenido por el alineamiento de Clustal Omega y el que posee más similitudes es el resultante del alineamiento progresivo dos a dos mediante la BLOSUM62.