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.

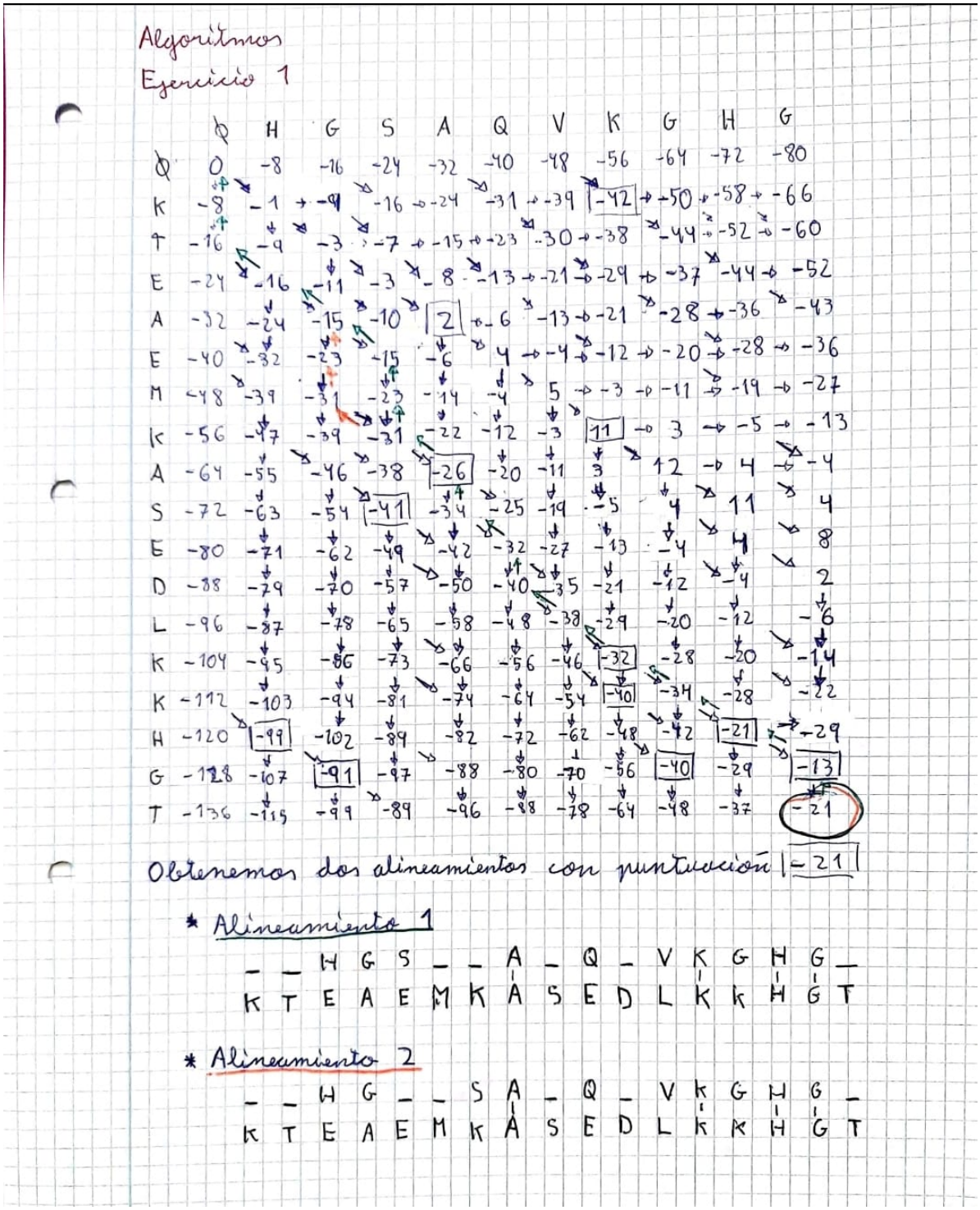


Figura 1: Matriz y alineamientos obtenidos.

Como se puede observar, La puntuación obtenida es -21 y se han encontrado dos alineamientos globales posibles.

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.

Métodos

Para este ejercicio decidimos utilizar EMBOSS Needle, Pairwise Sequence Alignment (PSA) para realizar los distintos alineamientos dos a dos; y Clustal Omega (1.2.4), Multiple Sequence Alignment (MSA) para el alineamiento múltiple.

Los alineamientos progresivos y guide trees correspondientes a los alineamientos dos a dos, los realizamos a mano.

Finalmente, obtuvimos los logos en R, utilizando los paquetes Biostrings, ggseqlogo y ggplot2.

Alineamientos dos a dos

Los parámetros que utilizamos para realizar este proceso son:

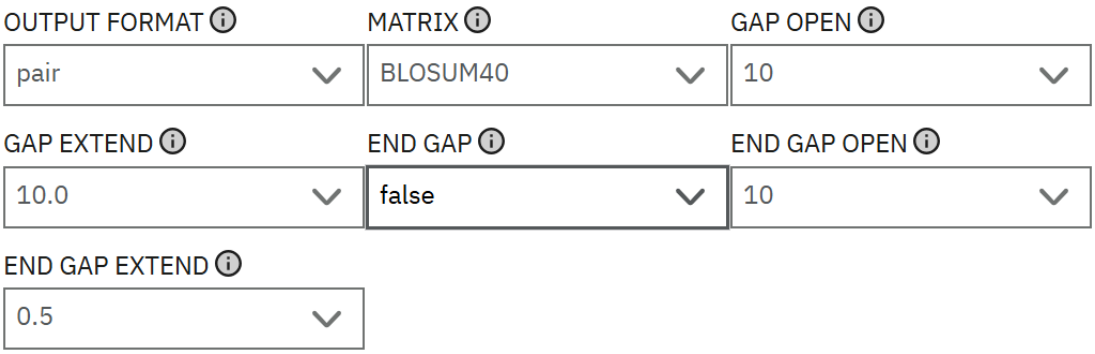


Figura 2: Parámetros para el alineamiento dos a dos.

Decidimos intentar mantener unos parámetros próximos a los utilizados en el ejercicio 1: una tabla BLOSUM40 y restar 10 por cada gap (lo más próximo a 8 que estaba disponible).

Obtuvimos los siguientes alineamientos son sus scores correspondientes, calculados dividiendo el número de coincidencias entre el número de residuos sin tener en cuenta los gaps:

```
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.

Y construímos la matriz de distancias, para la cual hacemos 1 - el score de similitud del alinemaiento:

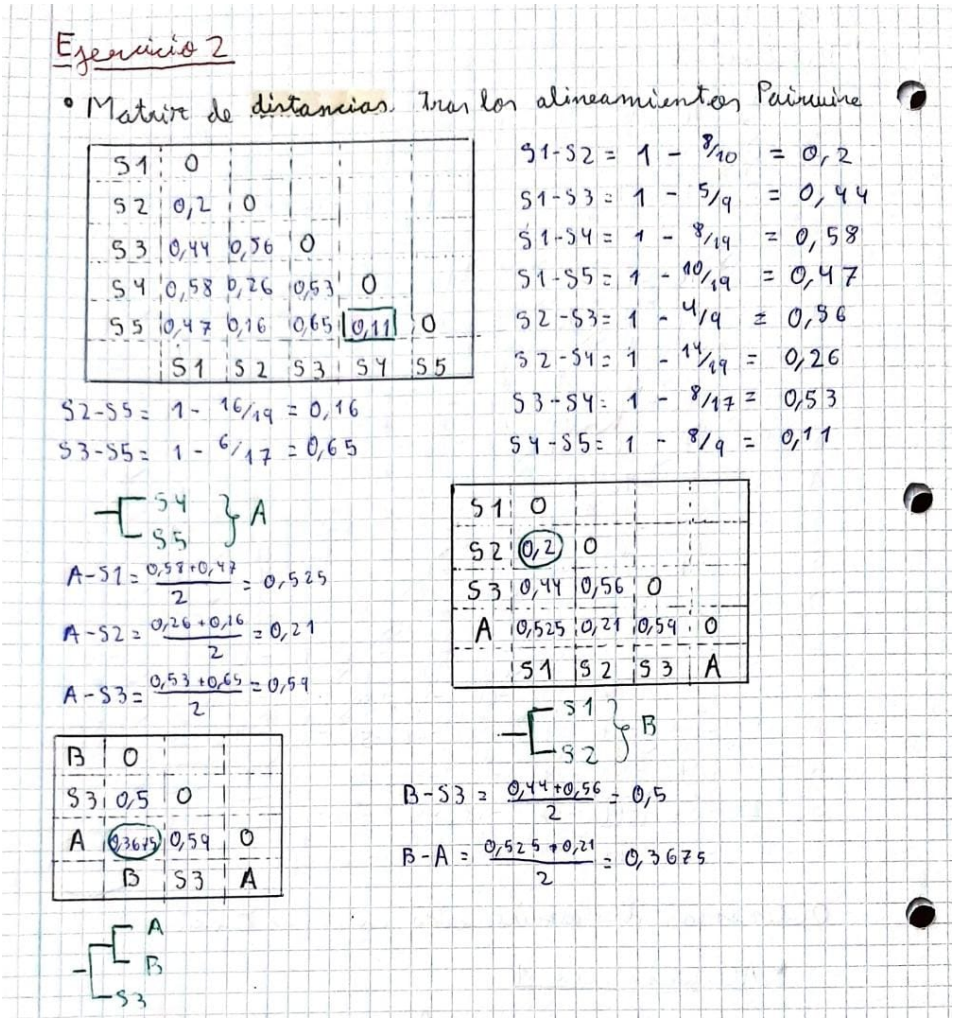


Figura 13: Construcción de matrices de distancias.

Finalmente, construímos el guide tree:

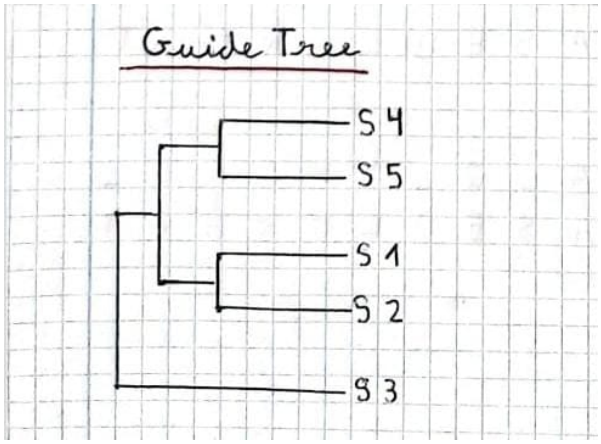


Figura 14: Guide Tree obtenido a partir de alineamientos 2 a 2 con parámetros similares al ejercicio 1.

Y el alineamiento progresivo:

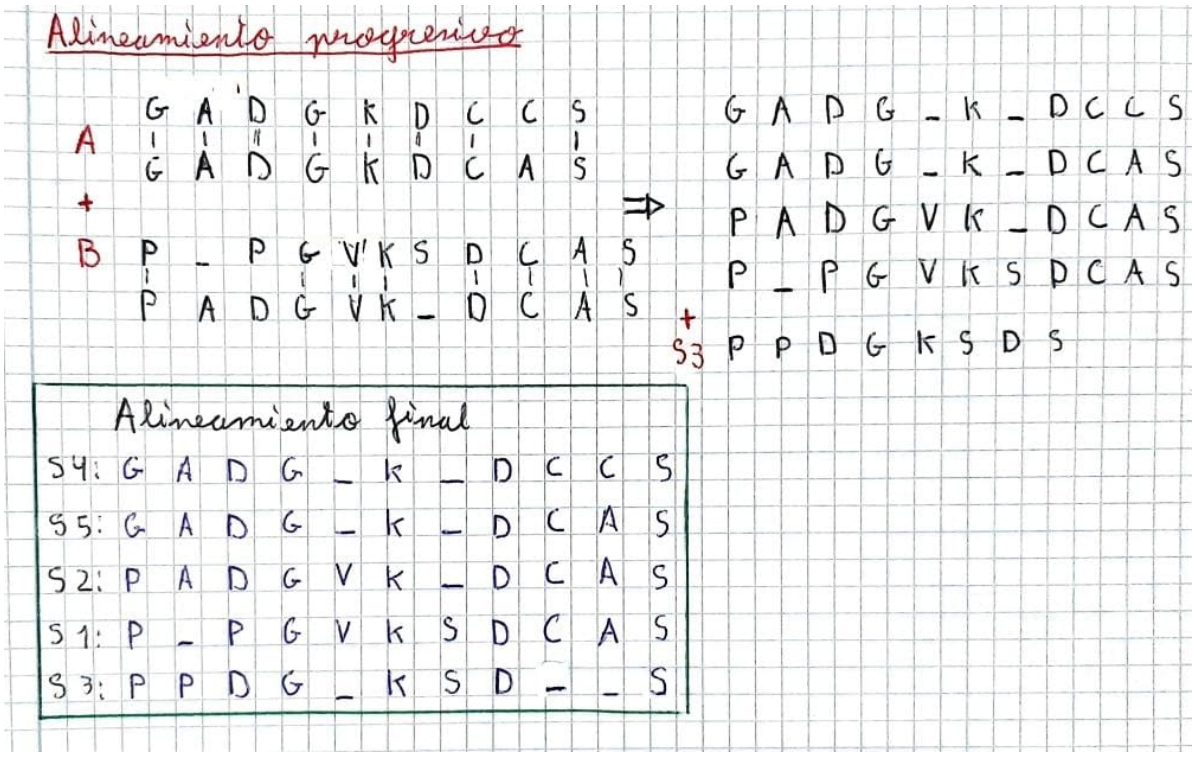


Figura 15: Alineamiento progresivo obtenido a partir de alineamientos 2 a 2 con parámetros similares al ejercicio 1.

Con parámetros por defecto

También, decidimos repetir este proceso pero dejando los parámetros que vienen por defecto, para comprobar si generaríamos el mismo alineamiento independientemente de los parámetros escogidos:

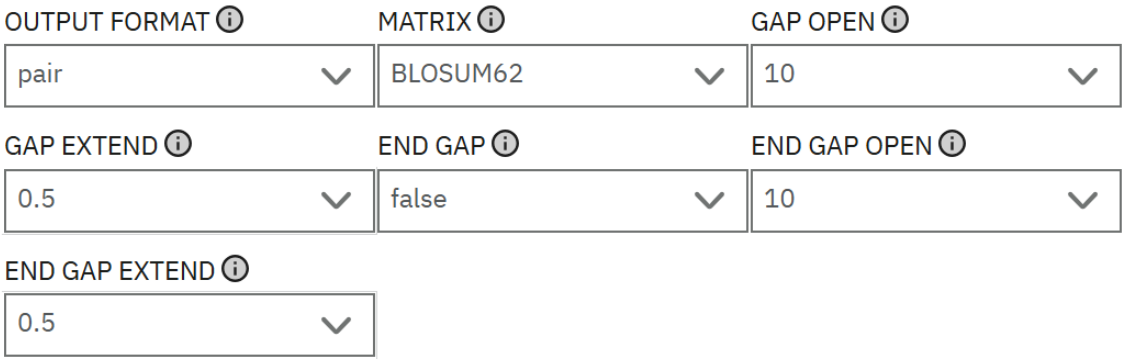


Figura 16: Parámetros para el alineamiento dos a dos por defecto.

Los alineamientos y scores obtenidos con estos parámetros fueron:

```
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.

La matriz de distancias correspondiente a estos alineamientos sería:

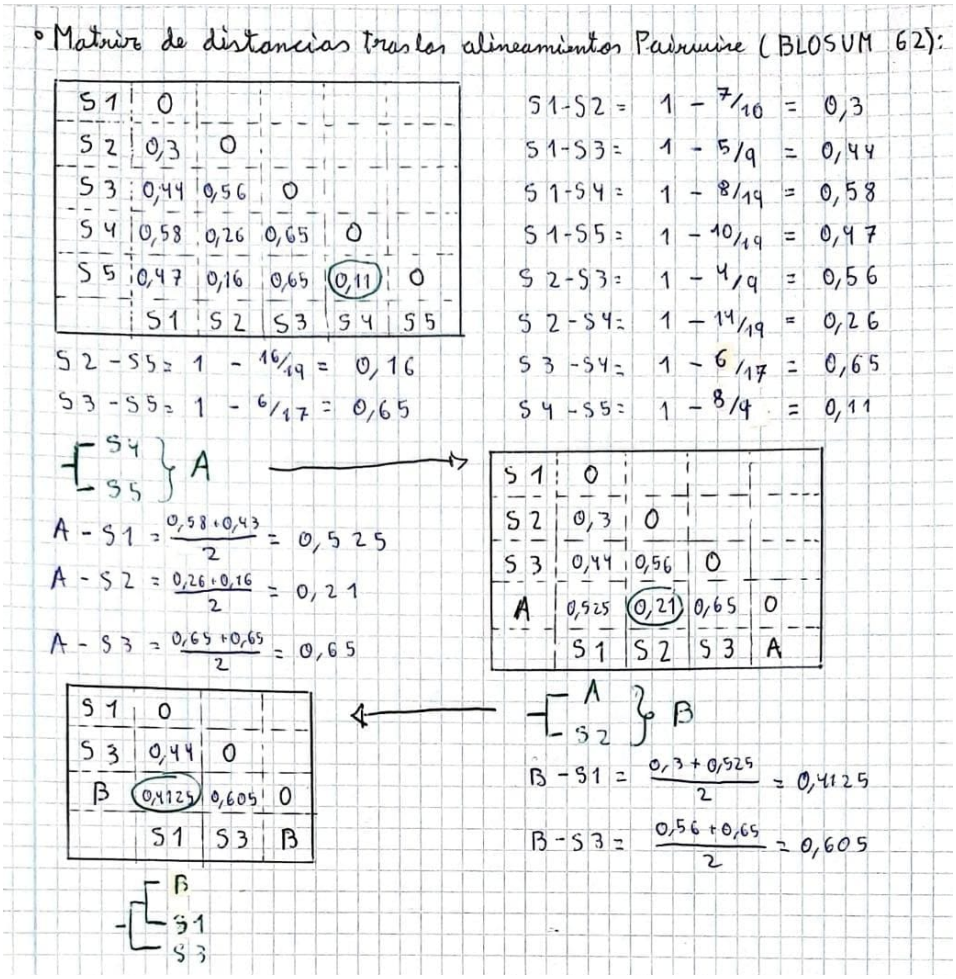


Figura 27: Construcción de matrices de distancias.

Finalmente, construímos el guide tree:

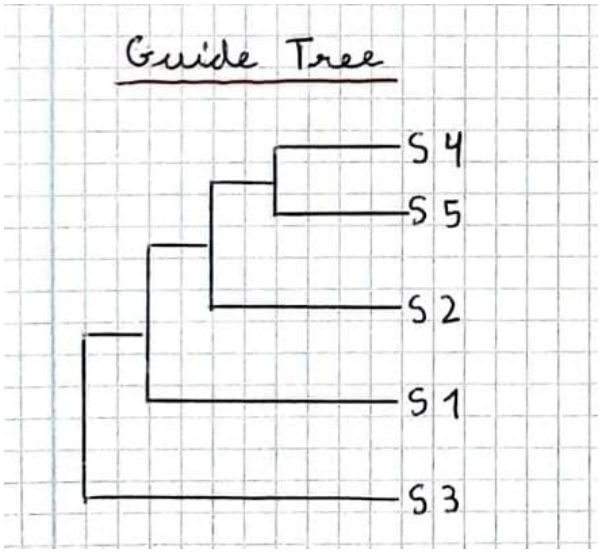


Figura 28: Guide Tree obtenido a partir de alineamientos 2 a 2

Y el alineamiento progresivo:

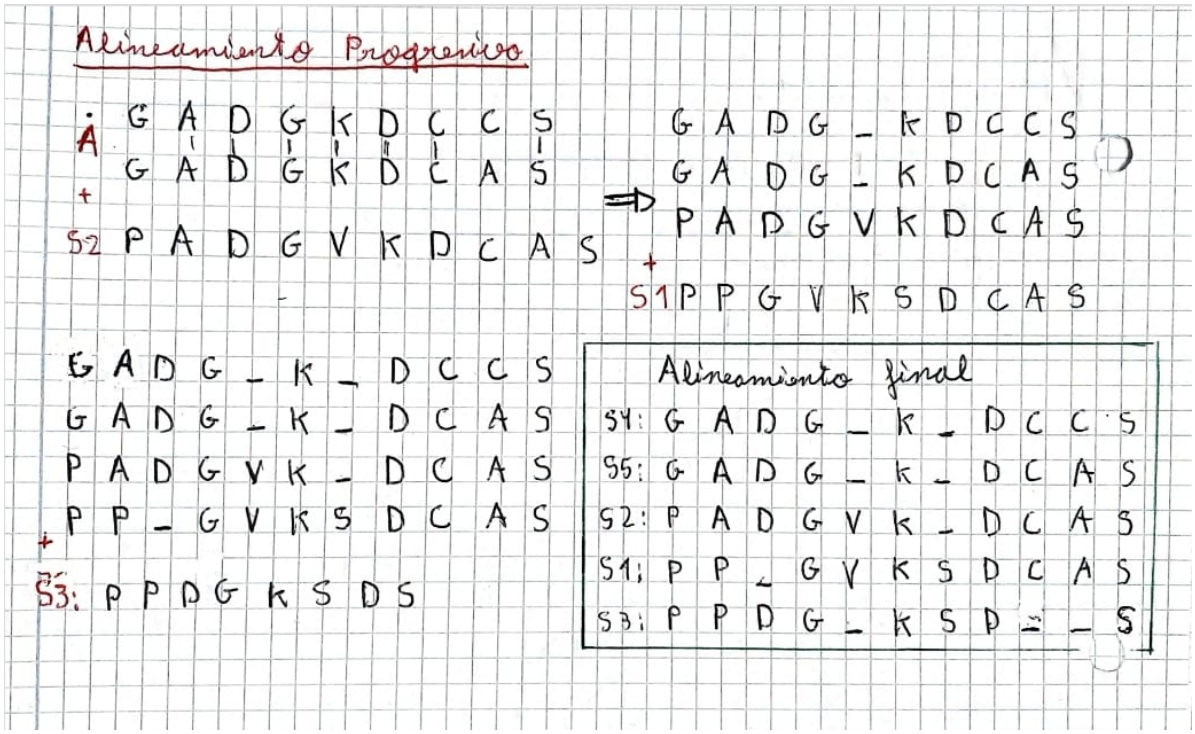


Figura 29: Alineamiento progresivo obtenido a partir de alineamientos 2 a 2 con matriz BLOSUM62

Alineamiento múltiple

Los parámetros utilizados para realizar este proceso fueron:

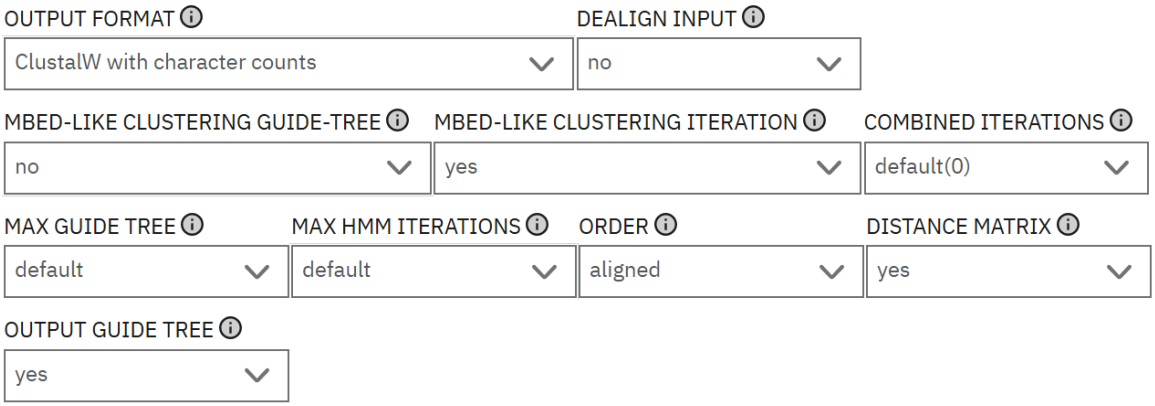


Figura 30: Parámetros para el alineamiento múltiple.

Obtivos el siguiente guide tree:

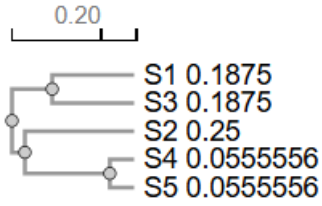


Figura 31: Guide tree del alineamiento múltiple por Clustal Omega.

Y este alineamiento múltiple:

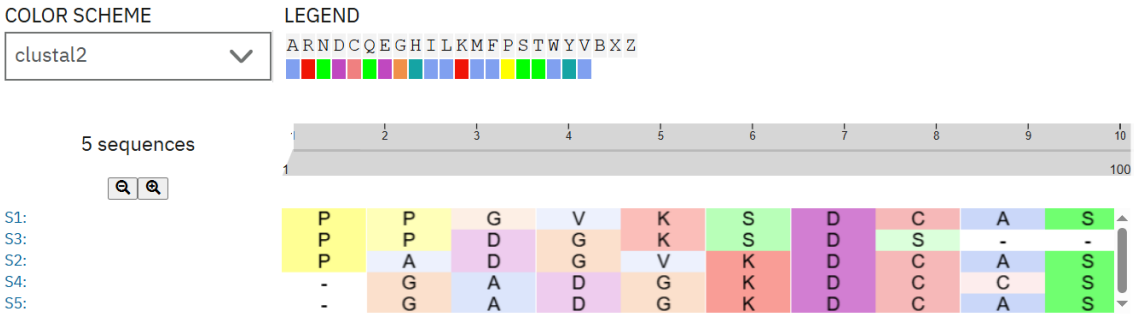


Figura 32: Alineamiento múltiple por Clustal Omega.

Comparación y logo

Observamos que los guide tree obtenidos por ambos alineamientos dos a dos, difieren entre sí y con el guide tree obtenido por el Clustal Omega; siendo común entre ellos únicamente el grupo S4-S5.

Respecto a los alineamientos progresivos obtenidos en los alineamientos 2 a 2 con la BLOSUM40 y BLOSUM62, son muy semejantes, variando únicamente por a posición de la segunda prolina (P) de la S1, debido a la utilización de distintas matrices de sustitución y los diferentes valores utilizados para penalizar los gaps.

Ambas varían mucho del alineamiento múltiple obtenido mediante el Clustal Omega, debido a que este método es heurístico y no garantiza encontrar el alineamiento globalmente óptimo, teniendo dificultad frente a deleciones e inserciones, en las que incluye gaps; siendo crucial 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.

Logo del alineamiento progresivo del alineamiento dos a dos mediante la BLOSUM40

Cargamos las librerías necesarias, cargamos los documentos FASTA de los alineamientos progresivos y obtuvimos el logo de ambos alineamientos dos a dos y del Clustal Omega.

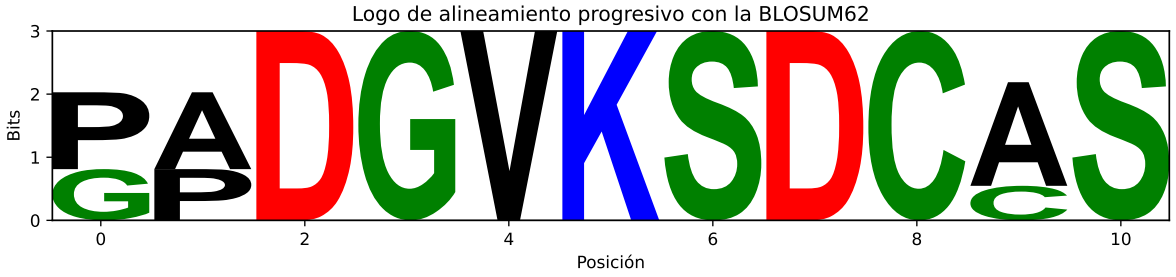
```
from Bio import AlignIO
import logomaker
import matplotlib.pyplot as plt
import numpy as np

# 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='-')

# Convertir a frecuencias relativas (ignorando gaps)
frequency_matrix = counts_matrix.div(counts_matrix.sum(axis=1), axis=0)

# Calcular información (en bits) para cada posición
information_matrix = logomaker.transform_matrix(frequency_matrix, \
  from_type='probability', to_type='information')

# Crear el logo
```

```
logo = logomaker.Logo(
    information_matrix,  # Usamos la matriz de información directamente
    color_scheme='chemistry',
    font_name='Arial',
    show_spines=True,
    stack_order='big_on_top'
)

# Limitar bits a 4.32 si es necesario
logo.ax.set_ylim(0, 4.32)

# Personalización adicional
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()
```

Logo del alineamiento progresivo del alineamiento dos a dos mediante la BLOSUM40

```
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_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='-')

# Convertir a frecuencias relativas (ignorando gaps)
frequency_matrix = counts_matrix.div(counts_matrix.sum(axis=1), axis=0)

# Calcular información (en bits) para cada posición
information_matrix = logomaker.transform_matrix(frequency_matrix, \
  from_type='probability', to_type='information')

# Crear el logo
logo = logomaker.Logo(
    information_matrix,  # Usamos la matriz de información directamente
    color_scheme='chemistry',
    font_name='Arial',
    show_spines=True,
    stack_order='big_on_top'
)

# Limitar bits a 4.32 si es necesario
logo.ax.set_ylim(0, 4.32)

# Personalización adicional
logo.ax.set_title("Logo de alineamiento progresivo con la BLOSUM62")
logo.ax.set_xlabel("Posición")
logo.ax.set_ylabel("Bits")
logo.ax.set_ylim(0, 3)

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



Logo del MSA por Clustal Omega

```
from Bio import AlignIO
import logomaker
import matplotlib.pyplot as plt
import numpy as np

# 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='-')

# Convertir a frecuencias relativas (ignorando gaps)
frequency_matrix = counts_matrix.div(counts_matrix.sum(axis=1), axis=0)

# Calcular información (en bits) para cada posición
information_matrix = logomaker.transform_matrix(frequency_matrix, \
  from_type='probability', to_type='information')

# Crear el logo
logo = logomaker.Logo(
    information_matrix,  # Usamos la matriz de información directamente
    color_scheme='chemistry',
    font_name='Arial',
    show_spines=True,
    stack_order='big_on_top'
)

# Limitar bits a 4.32 si es necesario
logo.ax.set_ylim(0, 4.32)

# Personalización adicional
logo.ax.set_title("Logo de alineamiento Clustal Omega")
logo.ax.set_xlabel("Posición")
logo.ax.set_ylabel("Bits")
logo.ax.set_ylim(0, 3)

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

De los tres logos obtenidos, podemos destacar que aquel con más variedad 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.