INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE COMPUTACIÓN

PROGRAMA DE MAESTRÍA

MEJORAS A LOS ALGORITMOS DE BIOINFORMÁTICA NEEDLEMAN-WUNSCH Y K-BAND MEDIANTE LA MEJORA DE LA MATRIZ DE SIMILITUD

TESIS PARA OPTAR AL GRADO DE

MAGISTER SCIENTIAE EN COMPUTACIÓN

ALONSO SABORÍO ARCE

CARTAGO, COSTA RICA

Resumen

Muchos algoritmos en el área de la Bioinformática se enfocan en métodos de alineamiento de secuencias; estos pueden llegar a tener problemas de manejo de memoria y / o espacio en disco. Por ejemplo, el algoritmo de Needleman-Wunsch logra alinear dos secuencias en su totalidad de forma óptima; sin embargo, se crea una matriz de tamaño $N \times M$, donde N y M son la cantidad de caracteres de las secuencias, la cual puede llegar a causar los problemas descritos.

El objetivo de esta tesis es plantear y demostrar una mejora al algoritmo de Needleman-Wunsch para aumentar su eficiencia. Con esto, una mezcla de dos algoritmos de programación dinámica junto con la minimización del uso de la matriz de similitud es suficiente para utilizar en menor cantidad el espacio en disco y / o memoria que el algoritmo mencionado.

La solución implementada es una variante del algoritmo de K-Band; el cual, consigue una solución óptima recorriendo la matriz una única vez, a diferencia del original que la recorre X cantidad de veces. La principal característica es expandir la banda en la matriz de similitud en momentos donde la alineación pueda convertirse en no óptima, es decir que se encuentre en el borde de la banda. Se generaron varios experimentos para validar que la solución implementada sea mejor que los algoritmos de Needleman-Wunsch y K-Band. Los resultados indican que la implementación hace un menor uso de la matriz de similitud y, al mismo tiempo, mejora el desempeño del espacio en disco y / o memoria en secuencias grandes.

Palabras clave: Needleman-Wunsch, K-Band, matriz de similitud, alineamiento de secuencias, Bioinformática

Abstract

Bioinformatics algorithms are limited by memory and disc management processes due to their large amount of data manipulation, like sequence alignment methods. For example, Needleman-Wunsch algorithm aligns two sequences optimally; it creates a matrix of *N* rows and *M* columns, where *N* and *M* are the amount of characters each sequence had. Based on the final size of the matrix these problems become restrictive.

This thesis's objective is to set and prove a Needleman-Wunsch algorithm improvement in order to enhance its efficiency. With the fusion of two dynamic programming algorithms and the minimization on the similarity matrix usage the amount of disc and / or memory used by the algorithm will be reduced.

The implemented solution is a modification of the K-Band algorithm, which gets the optimal solution parsing the matrix once, differing from its original version that parses it many times. The main feature of the new algorithm is how the band is expanded when the alignment cannot be optimal, that will happen when the parsing is in the border of the band. Some experiments were developed and ran in order to validate the solution created. With the results obtained, it was proved that the proposed improvement of the Needleman-Wunsch algorithm uses the similarity matrix less and improves the memory and disc performance on big sequences.

Keywords: Needleman-Wunsch, K-Band, similarity matrix, sequence alignment, Bioinformatics

Aprobación de Borrador de Tesis

Mejoras a los algoritmos de bioinformática Needleman-Wunsch y K-Band mediante la mejora de la matriz de similitud

Eddy Ramírez Jiménez, M. Sc.

Profesor Asesor (Visto Bueno)

Roberto Cortés Morales, Ph. D.

Director Maestría (Refrendo)

www.tec.ac.cr

TEC | Tecnológico de Costa Rica
Programa de Maestría
en Computación
Teléfono: 2550-2402

APROBACIÓN DE LA TESIS

"Optimización de la Matriz de Similitud del Algoritmo de Needleman-Wunsch"

TRIBUNAL EXAMINADOR

M.Sc. Eddy Ramírez Jiménez
Profesor Asesor

Ph.D. Carlos González Alvarado
Profesor Lector

M.Sc. Rafael Monge Montenegro
Profesor Externo

Dr. Roberto Cortés Morales
Coordinador del Programa
de Maestría en Computación

Junio, 2015

Dedicatoria

Quiero dedicar esta tesis, en primer lugar, a mi esposa y a mi hija; quienes han estado apoyándome y soportándome durante el transcurso de la maestría. De no ser por ellas esta tesis no tendría el mismo fruto.

Además quiero agradecer a los profesores que me apoyaron en este camino. Al profesor Carlos González que desde el principio me incitó a seguir adelante con el proyecto aunque pareciera imposible o difícil. Al profesor Eddy Ramírez por el apoyo incondicional brindado para con mi persona como profesor tutor y por las ideas y guías brindadas para poder llegar a concluir con esta tesis.

A mis padres, por estar siempre pendientes de mi avance y formación. Siempre dedicando unas palabras de motivación para seguir adelante y no caer al final.

Capitulo 1 Tabla de Contenidos

Índice de Figuras

Índice de Tablas

Índice de Fórmulas

Capitulo 1 Planteamiento del problema

1.1 Descripción informal del problema

Los algoritmos de *Alineamiento* son de gran importancia hoy en día por lo que su mejora puede significar una ayuda a la comunidad científica. Se puede denotar que existen muchos tipos diferentes de algoritmos de alineamiento, los cuales son mejoras unos de otros, en cuanto a rendimiento de espacio o disco, o nuevas formas de alinear las secuencias (Needleman & Wunsch, 1970). En su gran mayoría, estos algoritmos generan una matriz de puntuación con el fin de encontrar similitudes entre las secuencias a tratar. Si se toma en cuenta que se está haciendo uso de una matriz para ingresar valores, que posteriormente se usarán para tomar decisiones, se puede suponer el problema que este puede causar a la hora de: 1) manejo de memoria, 2) manejo de espacio en disco; considerando la magnitud de los datos que se va a llegar a tratar, proponiendo un problema de tipo $O_{(n)}$ (Backofen, 2011).

Un ejemplo muy simple se puede describir de la siguiente forma: al tratar de alinear completamente dos secuencias de ADN, ARN o proteína, se estaría generando una matriz de N x M en donde N es la cantidad de valores en la primer secuencia y M la cantidad de valores de la segunda secuencia (Backofen, 2011). Si lo ve desde un punto de vista técnico, se puede llegar a tener problemas grandes de memoria o espacio en disco (dependiendo de la implementación que se haya realizado). Con lo antes mencionado, el acceso a la matriz puede suponer un costo elevado debido a la cantidad de espacio utilizado NxM, ya sea para ingresar o consultar los datos; adicionalmente esto puede causar que el manejo de la matriz sea lento y complejo al mismo tiempo.

El área de la Bioinformática crece rápidamente (Transparency Market Research, 2012), por lo que cada uno de los descubrimientos o mejoras en algoritmos de la misma que se realicen computacionalmente debe mantenerse en una mejora continua; con el fin de que las respuestas a los problemas planteados por dicha área sigan siendo eficientes y eficaces. En la actualidad, cabe resaltar un algoritmo de gran importancia denominado *Alineamiento de secuencias*; la función del mismo es representar y comparar dos o más secuencias de ADN, ARN o proteínas para así resaltar las zonas comunes, éstas podrían ser de gran valor para indicar relaciones funcionales o evolutivas entre las secuencias.

Uno de los algoritmos más famosos de *Alineamiento de secuencias* es el algoritmo de *Needleman-Wunsch*. El mismo es conocido por generar el alineamiento óptimo para dos secuencias; sin embargo posee los problemas generales mencionados anteriormente; este cálculo puede generar un problema del tipo $O_{(nm)}$ (Backofen, 2011). Con este hecho, se puede observar la complejidad que se afronta en la actualidad con algoritmos óptimos aplicados a un volumen de datos que no pueda ser manejado por un computador.

En concreto, el problema por resolver es cómo mejorar el rendimiento que presenta el algoritmo de Needleman-Wunsch, referente a memoria o espacio en disco, de tal forma que se pueda procesar cadenas de mayor tamaño. Para lograr esto, debe de alterarse el manejo actual de la matriz de puntuación.

Este problema a primera vista parece ser equivalente a una modificación simple de manejo de matrices, pero en la práctica es completamente distinto. No basta con modificar únicamente el manejo de la matriz ya que se puede disminuir la respuesta óptima del algoritmo en *X* porcentaje (Rodriguez Tello, 2013). Por lo tanto, como se ejemplifica posteriormente, hay decenas de algoritmos de alineamiento que pueden mejorar el manejo de la matriz de puntuación pero sacrifican la respuesta óptima que *Needleman-Wunsch* logran.

1.2 Objetivos

1.2.1 Objetivo principal

Plantear y demostrar una mejora al algoritmo de Needleman-Wunsch para aumentar su eficiencia.

1.2.2 Objetivos específicos

1. Plantear y realizar ajustes al algoritmo para mejorar el rendimiento del mismo.
2. Validar el algoritmo obtenido contra secuencias ya probadas en dicho algoritmo para confirmar la validez de las respuestas obtenidas por el algoritmo generado.
3. Demostrar mediante un Benchmark que los ajustes realizados al algoritmo mejoran el rendimiento del mismo.

1.3 Hipótesis

Dada la explicación del tema anterior se puede denotar que uno de los grandes problemas de este algoritmo es el consumo desmesurado de memoria y espacio en disco; todo esto causado por la matriz de similitud que este genera, para conseguir una respuesta óptima al problema planteado. Por lo que se plantea la siguiente hipótesis:

La mezcla de dos algoritmos de programación dinámica junto con la minimización del uso de la matriz de similitud es suficiente para reducir la complejidad espacial que presenta el algoritmo de Needleman-Wunsch.

1.4 Justificación

El enfoque principal de este proyecto es el Algoritmo de Needleman-Wunsch. Este algoritmo genera un alineamiento óptimo, pero tiene como problema el uso de una o varias matrices con diversos resultados para realizar la toma de decisiones. Cabe destacar que al realizar cualquier tipo de análisis en gran volumen de datos sobre una matriz, puede generar problemas de complejidad temporal y espacial. Lo que se desea realizar es una mejora en la forma de utilizar la matriz en este algoritmo. En el mejor de los casos, se desea que la mejora de este algoritmo cause un mejor desempeño en la ejecución y respuesta del problema, por lo que el mismo podría pasar de ser un algoritmo teórico a uno práctico, usado por informáticos, biólogos, y otros grupos de investigadores.

Capitulo 2 Marco Teórico

El problema planteado se puede encontrar en muchos algoritmos que son usados hoy en día, los cuales son variantes del Needleman-Wunsch. Dada la importancia de este tipo de algoritmos para un sector de la ciencia e investigación, hay una cantidad de algoritmos que llevan a cabo la acción de alinear secuencias (Pinzon, 2010). Sin embargo, ninguno logra obtener una alineación óptima evitando los problemas que se han mencionado anteriormente (Rodriguez Tello, 2013).

Algunos de los algoritmos similares, o que de alguna forma resultan para la realización del proyecto, se presentan en las siguientes secciones:

2.1 Needleman-Wunsch

Este algoritmo es usado en alineamientos de tipo globales únicamente para dos secuencias. El mismo fue creado en 1970 por Saul Needleman y Christian Wunsch. Este algoritmo se basa en el principio de la programación dinámica. (Wunsch & Needleman, 1970)

El problema que tiene este algoritmo es el uso de espacio en disco. Bien es sabido que este tipo de algoritmos son empleados para alinear secuencias sumamente largas, como es el caso del genoma humano con 3.000 millones de pares de bases (NHGRI, 2011), por lo que el espacio en disco es de suma importancia (Santamaria, 2013).

Siguiendo con lo comentado anteriormente, se ha demostrado que este algoritmo es de orden $O_{(nm)}$ tanto en su tiempo de ejecución como en el espacio necesario para ejecución (Backofen, 2011).

2.1.1 Explicación del Algoritmo (Needleman & Wunsch, 1970)

Se procederá a explicar paso a paso el algoritmo mencionado. Se empezará comentando los elementos de entrada del algoritmo, luego sus elementos de salida y posteriormente se ofrecerá una breve explicación de lo que realiza el algoritmo en sí.

Los elementos de entrada que se necesitan para que este algoritmo de alineamiento global de dos secuencias funcione son:

- Un alfabeto *A* que tenga los símbolos a usar en la secuencia.
- Dos cadenas de texto *X* y *Y* cuyo tamaño sea *m* y *n* respectivamente.

    - Sea $X[0 \ldots m]$ la primera cadena a alinear.
        - *X [0]* es el primer carácter.
        - *X [m]* es el último carácter.
    - Sea *Y [0…n]* la segunda cadena a alinear.
        - *Y [0]* es el primer carácter.
        - *Y [n]* es el último carácter.
- Función $S(X[i], Y[j])$
    - X [i], Y [j] existen en el alfabeto *A*.
    - Esta función dará un puntaje de similitud de la letra X [i] con la letra Y [j].
- Un valor *d* que especifica un puntaje a la operación de no emparejar una letra de las secuencias con otra (Wunsch & Needleman, 1970).

Los elementos de salida para este algoritmo son:

- Puntaje máximo que se logra al alinear las secuencias *X* y *Y*.
- El alineamiento que da el puntaje máximo y óptimo para las dos secuencias dadas.

El algoritmo va a recibir dos cadenas de texto *X* y *Y*. Seguidamente, el mismo irá construyendo una matriz *F* y cada uno de sus elementos *F [i, j]* será el valor para el alineamiento óptimo de *X [0…i]* y *Y [0…j]*.

Para iniciar se debe de inicializar el valor de la matriz *F [0, 0]* con el valor de *0*.

Seguidamente se empieza a llenar la matriz desde la esquina superior izquierda hasta la esquina inferior derecha. Para conseguir el valor de *F [i, j]* han de estar debidamente llenos los valores de:

- *F [i - 1, j - 1]*
- *F [i – 1, j]*
- *F [i, j - 1]*

Cabe destacar que cada una de estas celdas debe de estar llena con sus valores de alineamiento óptimos. Teniendo el conocimiento de estos tres valores se pueden tener tres posibles caminos para completar el alineamiento de *X [1…i]* y *Y [1…j]*. Se pueden alinear de la siguiente forma:

- *X[i]* con *Y[j]*
- *X[i]* con un gap
- *Y[j]* con un gap

La forma recursiva es la siguiente:

$$F[i,j] = max \begin{cases} F[i-1, j-1] + s(x[i], y[j]) \\ F[i-1, j] - d \\ F[i, j-1] - d \end{cases}$$

Fórmula 1 – Valor de la mejor celda para el alineamiento óptimo

Si nos posicionamos en el lado superior izquierdo de la matriz, se deben especificar los valores de *F [i, 0]* y *F [0, j]*.

- El valor de *F [i, 0]* representa el prefijo de *X* como un gap, por lo que se define *F [i, 0] = i * d*.
- El valor de *F [0, j]* representa el prefijo de *Y* como un gap, por lo que se define *F [0, j] = j * d*.

Como este algoritmo desea obtener el valor y alineamiento óptimo, se va a guardar en cada celda de la matriz un *puntero* el cual puede tener los siguientes valores:

- *Izquierda*
- *Arriba*
- *Diagonal*

Estos valores pueden tenerse en una matriz separada, o en la misma matriz *F*, la cual debe tener el mismo tamaño que la matriz *F*. Los valores insertados indican cuál de estas tres opciones se usaron para realizar el valor de *F [i, j]*. Si fuera el caso en donde dos opciones dan como resultado el mismo valor *F [i, j]*, se escoge cualquier valor arbitrariamente. Si se deseara saber todos los posibles alineamientos óptimos, se pueden retener todos los valores mencionados anteriormente en la matriz para su futura consulta.

Al finalizar el llenado de la matriz *F*, se procede a leer el primer valor localizado en la esquina inferior derecha *F [n, m]*. Cabe denotar que *F [n, m]* no tiene el máximo valor en todo el lado derecho o todo el lado inferior de la matriz *F*, sin embargo al iniciar en este punto significa omitir uno o más caracteres del final de una de las hileras.

Para devolver las nuevas hileras ya alineadas, se deben crear dos nuevas hileras *u* y *v*:

- *u* es la hilera creada a partir de *x.*
- *v* la hilera creada a partir de *y*.
- Dichas variables se inicializan como *u = x[n]* y *v=y[m]*.

Por último se utiliza el proceso de *trace-back*. Es decir, se siguen los punteros anteriormente mencionados desde la celda con el máximo valor y dependiendo del valor obtenido así será el valor a agregar en *u* y *v*:

- *Diagonal*: agregar un carácter a cada hilera (a la izquierda).
- *Arriba:* agregar un gap a *u* y un carácter a la izquierda de *v*.
- *Izquierda:* agregar un gap a *v* y un carácter a la izquierda de *u* .

2.1.1.1 Ejemplo del algoritmo de Needleman-Wunsch

Para este ejemplo únicamente se utiliza la matriz *F* y en ella se irán agregando los *punteros*. Las flechas, dependiendo del color significan lo siguiente:

- **Azul:** puntero recién insertado en la tabla o camino tomado por el *trace back.*
- **Anaranjado:** puntero insertado anteriormente.

Tomando en cuenta que se tienen las siguientes condiciones:

- **Alfabeto:** {A, T, C, G}
- **Hilera 1:** ATCG
- **Hilera 2:** TCG
- **Gap:** -1
- **Pareja:** 1
- **Diferentes:** -1

Se puede generar una matriz de sustitución *S* para el alfabeto mencionado, el cual sería el siguiente:

	A	***T***	***C***	***G***
A	1	-1	-1	-1
T	-1	1	-1	-1
C	-1	-1	1	-1
G	-1	-1	-1	1

Figura 1. Ejemplo de matriz de sustitución

Seguidamente se procede a llenar la matriz *F* con los valores de los gaps (*F [i, 0]*, *F [0, j]*):

	-	***T***	***C***	***G***
-	0	-1	-2	-3
A	-1			
T	-2			
C	-3			
G	-4			

Figura 2. Ejemplo de llenado de matriz de puntuación 1 de 14

Si se calcula el valor de *F [1, 1]*, se obtienen los siguientes valores:

$$F[1,1] = max\begin{cases} F[0,0] + s(x[1], y[1]) = -1 \\ F[0,1] - 1 = -2 \\ F[1,0] - 1 = -2 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	***T***	***C***	***G***
-	0	-1	-2	-3
A	-1	-1		
T	-2			
C	-3			
G	-4			

Figura 3. Ejemplo de llenado de matriz de puntuación 2 de 14

Si se calcula el valor de *F [1, 2]*, se obtienen los siguientes valores:

$$F[1,2] = max\begin{cases} F[0,1] + s(x[1], y[2]) = 0 \\ F[0,2] - 1 = -3 \\ F[1,1] - 1 = -2 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	***T***	***C***	***G***
-	0	-1	-2	-3
A	-1	-1		
T	-2	0		
C	-3			
G	-4			

Figura 4. Ejemplo de llenado de matriz de puntuación 3 de 14

Si se calcula el valor de *F [1, 3]*, se obtienen los siguientes valores:

$$F[1,3] = max\begin{cases} F[0,2] + s(x[1], y[3]) = -3 \\ F[0,3] - 1 = -4 \\ F[1,2] - 1 = -1 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	*T*	*C*	*G*
-	0	-1	-2	-3
A	-1	-1		
T	-2	0		
C	-3	-1		
G	-4			

Figura 5. Ejemplo de llenado de matriz de puntuación 4 de 14

Si se calcula el valor de *F [1, 4]*, se obtienen los siguientes valores:

$$F[1,4] = max\begin{cases} F[0,3] + s(x[1], y[4]) = -4 \\ F[0,4] - 1 = -5 \\ F[1,3] - 1 = -2 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	*T*	*C*	*G*
-	0	-1	-2	-3
A	-1	-1		
T	-2	0		
C	-3	-1		
G	-4	-2		

Figura 6. Ejemplo de llenado de matriz de puntuación 5 de 14

Si se calcula el valor de *F [2, 1]*, se obtienen los siguientes valores:

$$F[2,1] = max\begin{cases} F[1,0] + s(x[2], y[1]) = -2 \\ F[1,1] - 1 = -2 \\ F[2,0] - 1 = -3 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	***T***	***C***	***G***
-	0	-1	-2	-3
A	-1	-1	-2	
T	-2	0		
C	-3	-1		
G	-4	-2		

Figura 7. Ejemplo de llenado de matriz de puntuación 6 de 14

Si se calcula el valor de *F [2, 2]*, se obtienen los siguientes valores:

$$F[2,2] = max\begin{cases} F[1,1] + s(x[2], y[2]) = -2 \\ F[1,2] - 1 = -1 \\ F[2,1] - 1 = -3 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	***T***	***C***	***G***
-	0	-1	-2	-3
A	-1	-1	-2	
T	-2	0	-1	
C	-3	-1		
G	-4	-2		

Figura 8. Ejemplo de llenado de matriz de puntuación 7 de 14

Si se calcula el valor de *F [2, 3]*, se obtienen los siguientes valores:

$$F[2,3] = max\begin{cases} F[1,2] + s(x[2], y[3]) = 1 \\ F[1,3] - 1 = -2 \\ F[2,2] - 1 = -2 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	*T*	*C*	*G*
-	0	-1	-2	-3
A	-1	-1	-2	
T	-2	0	-1	
C	-3	-1	1	
G	-4	-2		

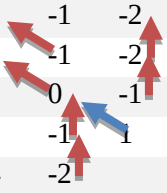

Figura 9. Ejemplo de llenado de matriz de puntuación 10 de 14

Si se calcula el valor de *F [2, 4]*, se obtienen los siguientes valores:

$$F[2,4] = max\begin{cases} F[1,3] + s(x[2], y[4]) = -2 \\ F[1,4] - 1 = -3 \\ F[2,3] - 1 = 0 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	*T*	*C*	*G*
-	0	-1	-2	-3
A	-1	-1	-2	
T	-2	0	-1	
C	-3	-1	1	
G	-4	-2	0	

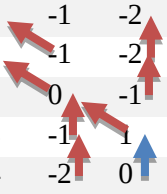

Figura 10. Ejemplo de llenado de matriz de puntuación 9 de 14

Si se calcula el valor de *F [3, 1]*, se obtienen los siguientes valores:

$$F[3,1] = max\begin{cases} F[2,0] + s(x[3], y[1]) = -3 \\ F[2,1] - 1 = -3 \\ F[3,0] - 1 = -4 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	*T*	*C*	*G*
-	0	-1	-2	-3
A	-1	-1	-2	-3
T	-2	0	-1	
C	-3	-1	1	
G	-4	-2	0	

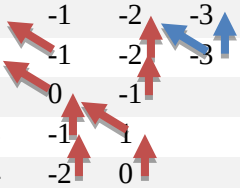

Figura 11. Ejemplo de llenado de matriz de puntuación 10 de 14

Si se calcula el valor de *F [3, 2]*, se obtienen los siguientes valores:

$$F[3,2] = max\begin{cases} F[2,1] + s(x[3], y[2]) = -3 \\ F[2,2] - 1 = -2 \\ F[3,1] - 1 = -4 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	*T*	*C*	*G*
-	0	-1	-2	-3
A	-1	-1	-2	-3
T	-2	0	-1	-2
C	-3	-1	1	
G	-4	-2	0	

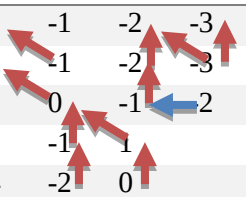

Figura 12. Ejemplo de llenado de matriz de puntuación 11 de 14

Si se calcula el valor de *F [3, 3]*, se obtienen los siguientes valores:

$$F[3,3] = max\begin{cases} F[2,2] + s(x[3], y[3]) = -2 \\ F[2,3] - 1 = 0 \\ F[3,2] - 1 = -3 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	*T*	*C*	*G*
-	0	-1	-2	-3
A	-1	-1	-2	-3
T	-2	0	-1	2
C	-3	-1	1	0
G	-4	-2	0	

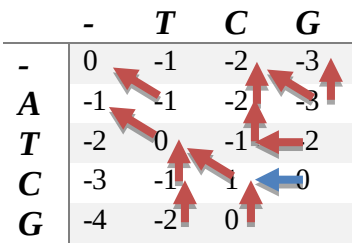

Figura 13. Ejemplo de llenado de matriz de puntuación 12 de 14

Si se calcula el valor de *F [3, 4]*, se obtienen los siguientes valores:

$$F[3,4] = max \begin{cases} F[2,3] + s(x[3], y[4]) = 2 \\ F[2,4] - 1 = -1 \\ F[3,3] - 1 = -1 \end{cases}$$

Por lo que la matriz quedaría de la siguiente manera:

	-	*T*	*C*	*G*
-	0	-1	-2	-3
A	-1	-1	-2	-3
T	-2	0	-1	2
C	-3	-1	1	0
G	-4	-2	0	2

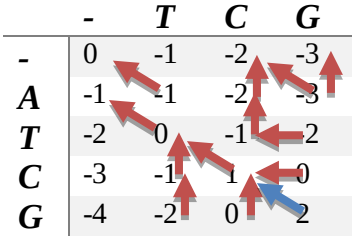

Figura 14. Ejemplo de llenado de matriz de puntuación 13 de 14

Una vez concluido el algoritmo para generar la matriz *F*, se prosigue con el *Trace back*. En azul se denotara el camino tomado por el algoritmo para generar el alineamiento óptimo.

	-	*T*	*C*	*G*
-	0	-1	-2	-3
A	-1	-1	-2	-3
T	-2	0	-1	2
C	-3	-1	1	0
G	-4	-2	0	2

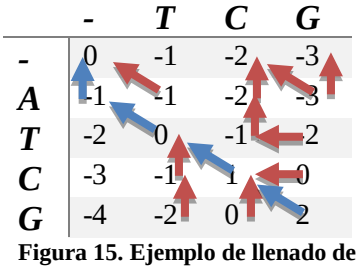

Figura 15. Ejemplo de llenado de matriz de puntuación 14 de 14

Con esto se puede realizar el alineamiento, el cual sería:

ATCG

*TCG

Con un valor de: -1 + 1 + 1 + 1 = 2

2.1.2 Ventajas y Desventajas del Algoritmo

La ventaja más notable del algoritmo de Needleman-Wunsch con respecto a cualquier otro algoritmo de alineamiento de secuencias global es que va a desplegar un alineamiento óptimo cada vez que se requiera. Con esto se puede asegurar que, al utilizar este sistema, el mismo indicará donde exactamente deben colocarse los valores de proteína, ADN, ARN para que queden alineadas las secuencias, ya sea con un gap o con un respectivo valor de las mismas (Huson, 2003).

Si se presta un poco más de atención al algoritmo se puede notar que utiliza un método de programación dinámica del tipo Top-Down (Xumari, 1967); esta metodología se basa en obtener la solución óptima al problema planteado mediante 3 pasos simples y esenciales:

1. Dividir el problema en sub-problemas
2. Resolver los sub-problemas de forma óptima dividiéndolos en más sub-problemas
3. Usar las soluciones óptimas de cada sub-problema para construir la solución óptima al problema original

Con estos 3 simples pasos se puede obtener la respuesta óptima, pero lo que hay que detallar es la forma en que esta respuesta puede llegar a verse afectada por la complejidad del problema. Se puede afirmar con los 3 pasos que conforme se va dividiendo el problema en sub-problemas se va a llegar a los escenarios simples, los cuales se resuelven y finalmente se obtiene un compilado de cada respuesta a los sub-problemas creados (Xumari, 1967).

Pero, ¿qué implica todo este proceso en realidad? El uso de mucho espacio en disco y consumo desmedido de memoria. ¿Cómo se puede probar esa afirmación? Con el hecho de saber que este algoritmo es de orden $O_{(nm)}$. Si se trata de ver desde el punto de vista del algoritmo, se puede notar que tiene que estar recorriendo cada una de las letras de un alineamiento y obtener el resultado con el match de la misma con todos los valores del otro alineamiento; con esto se obtiene el valor del match (esto es lo mismo que dividir el problema en muchos sub-problemas). Al mismo tiempo, se puede observar que hace un recorrido lineal del largo de la cadena 1 contra el largo de la cadena 2; haciendo el problema del orden anteriormente citado.

2.2 Hirschberg

Este algoritmo fue creado por Dan Hirschberg, al igual que el algoritmo de Needleman-Wunsch se base en el principio de la programación dinámica. Este algoritmo utiliza la *distancia de Levenshtein*, definida por la suma de los costos de: inserción, reemplazo, eliminación y acciones nulas necesarias para cambiar una secuencia en otra. Se puede mencionar que es una versión de *divide y vencerás*. (Hirschberg, 1975)

Se sabe que el algoritmo de Needleman-Wunsch tiene un orden de $O_{(nm)}$ en tiempo y espacio; mientras que el de Hirschberg tiene $O_{(nm)}$ en tiempo pero $O_{(\min\{nm\})}$ en espacio.

2.2.1 Explicación del Algoritmo (Hirschberg, 1975)

Se tiene el carácter *X [i]* donde $1 < i \leq largo(X)$.

X y *Y* son secuencias para linear. Suponiendo que *x* es un caracter de *X*, y *y* es un caracter de *Y*.

Se define la función *NWScore(X, Y)*. Esta rutina retorna la última línea de la matriz de valores de Needleman-Wunsch. El pseudo- código de esta función se puede ver de la siguiente manera:

```
function NWScore(X,Y)
   Score(0,0) = 0
   for j=1 to length(Y)
     Score(0,j) = Score(0,j-1) + Ins(Yj)
   for i=1 to length(X)
     Score(i,0) = Score(i-1,0) + Del(Xi)
     for j=1 to length(Y)
       scoreSub = Score(i-1,j-1) + Sub(Xi, Yj)
       scoreDel = Score(i-1,j) + Del(Xi)
       scoreIns = Score(i,j-1) + Ins(Yj)
       Score(i,j) = max(scoreSub, scoreDel, scoreIns)
     end
   end
   for j=0 to length(Y)
     LastLine(j) = Score(length(X),j)
   return LastLine
```

Figura 16. Pseudocódigo de la función NWScore.

En cualquier momento se puede notar que *NWScore* solo requiere de las 2 últimas filas de la matriz de valores; por lo que esta función puede ser implementada $O(\min\{largo(x), largo(y)\})$ en espacio. El pseudocódigo del algoritmo de Hirschberg es el siguiente:

```
function Hirschberg(X,Y)
   Z = ""
   W = ""
   if length(X) == 0 or length(Y) == 0
     if length(X) == 0
       for i=1 to length(Y)
         Z = Z + '-'
         W = W + Y_i
       end
     else if length(Y) == 0
       for i=1 to length(X)
         Z = Z + X_i
         W = W + '-'
       end
     end
   else if length(X) == 1 or length(Y) == 1
     (Z,W) = (Z,W) + NeedlemanWunsch(X,Y)
   else
     xlen = length(X)
     xmid = length(X)/2
     ylen = length(Y)

     ScoreL = NWScore(X_1:xmid, Y)
     ScoreR = NWScore(rev(X_xmid+1:xlen), rev(Y))
     ymid = PartitionY(ScoreL, ScoreR)

     (Z,W) = (Z,W) + Hirschberg(X_1:xmid, y_1:ymid)
     (Z,W) = (Z,W) + Hirschberg(X_xmid+1:xlen, Y_ymid+1:ylen)
   end
   return (Z,W)
```

Figura 17. Pseudocódigo de la function de Hirschberg

2.2.1.1 Ejemplo del algoritmo de Hirschberg

Los valores de entrada son los siguientes:

- **X:** AGTACGCA
- **Y:** TATGC
- **Del(x):** -2
- **Ins(y):** -2
- **Sub(x, y) =** $\boldsymbol{Sub(x,y)}: \begin{cases} \boldsymbol{+2, if\ x=y} \\ \boldsymbol{-1, if\ x \neq y} \end{cases}$

El valor optimo seria:

W = AGTACGCA

Z = --TATGC-

Cuando se llama a la función principal de Hirschberg, se produce la llamada al *NWScore* con parte de *X* y se crearía la siguiente matriz:

	-	*T*	*A*	*T*	*G*	*C*
-	0	-2	-4	-6	-8	-10
A	-2	-1	0	-2	-4	-6
G	-4	-3	-2	-1	0	-2
T	-6	-2	-4	0	-2	-1
A	-8	-4	0	2	-1	-3

Figura 18. Matriz de similitud del algoritmo de Hirschberg con la primer parte de X

Seguidamente, si se realiza *NWScore (rev (CGCA), rev (Y))*

	-	*C*	*G*	*T*	*A*	*T*
-	0	-2	-4	-6	-8	-10
A	-2	-1	-3	-5	-4	-6
C	-4	0	-2	-4	-6	-5
G	-6	-2	2	0	-2	-4
C	-8	-4	0	1	-1	-3

Figura 19. Matriz de similitud del algoritmo de Hirschberg con la segunda parte de X

Por lo tanto las últimas líneas respectivamente son:

ScoreL = [-8 -4 0 -2 -1 -3]

ScoreR = [-8 -4 0 1 -1 -3]

Se puede también visualizar este algoritmo en forma de árbol. Cabe destacar que las hojas del mismo son la respuesta al algoritmo.

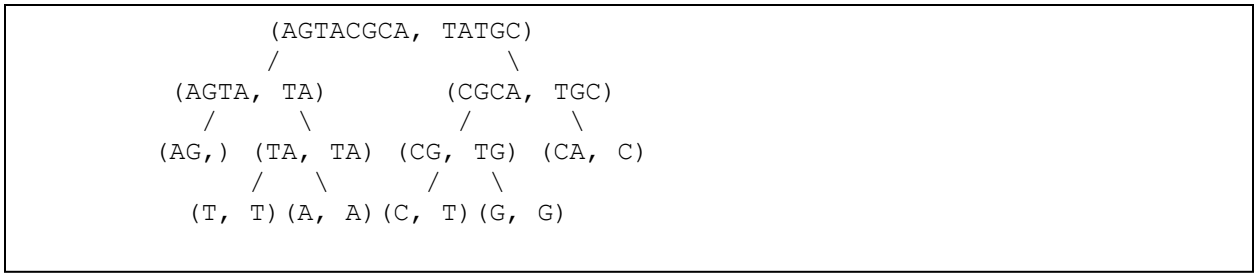

Figura 20. Algoritmo de Hirschberg en forma gráfica (Hirschberg, 1975)

2.3 K-Band

2.3.1 Explicación del Algoritmo

Este algoritmo es conocido como una alteración al algoritmo de Needleman-Wunsch; lo que hace es disminuir la cantidad de celdas utilizadas en la matriz de similitud. Por ejemplo: si tenemos una matriz de tamaño *NxM*, el algoritmo de Needleman-Wunsch utiliza todas las celdas de la matriz llenándolas de valores para obtener el resultado de la alineación, mientras que el algoritmo de K-Band utiliza únicamente las celdas que se encuentren dentro de las bandas *K* escogidas (Jackson & Aluru, 2001).

Tomando en cuenta lo anteriormente mencionado, podemos empezar a pensar la razón de porque alguien querría alinear secuencias con un sistema de bandas; es decir, que la matriz de similitud no se encuentre totalmente llena. La respuesta a esto es que por lo general los algoritmos como K-Band son usados para alinear secuencias muy parecidas, tanto en tamaño como en especie. La idea de este algoritmo es restringir el uso de la matriz de similitud en dos bandas de tamaño *K*; esperando que las secuencias no cambien mucho para obtener un alineamiento óptimo. Lo que se espera es que el alineamiento óptimo se encuentre en la diagonal principal. Por ejemplo este algoritmo es usado para (Huson, 2003):

1. Alinear secuencias de la misma especie con el paso del tiempo para poder visualizar puntualmente si la especie sufrió de alguna mutación o cambio con el paso del tiempo.
2. En ciencia forense averiguar si el ADN obtenido en el cuerpo de una víctima es igual a un conjunto de muestras, con el fin de encontrar al culpable. Si se demuestra que la alineación difiere en más de un par de gaps se puede demostrar que no son iguales por lo que se procede a evaluar otra muestra.

Se puede medir la velocidad del algoritmo de la siguiente forma: las dos secuencias usadas son de igual tamaño *NxN*. Dado esto, el algoritmo tendrá en el mejor de los casos una velocidad de $O(nk)$ mientras que en el peor de los casos seria de $O(n^2)$ (Jackson & Aluru, 2001).

El algoritmo se puede ejemplificar de una manera simple y fácil para poder exponer la utilidad del mismo. Supongamos que tenemos dos secuencias *A* y *B*, que tienen el mismo tamaño *N* y un mismo valor de *K = 4*. Esto, se puede representar de la siguiente forma:

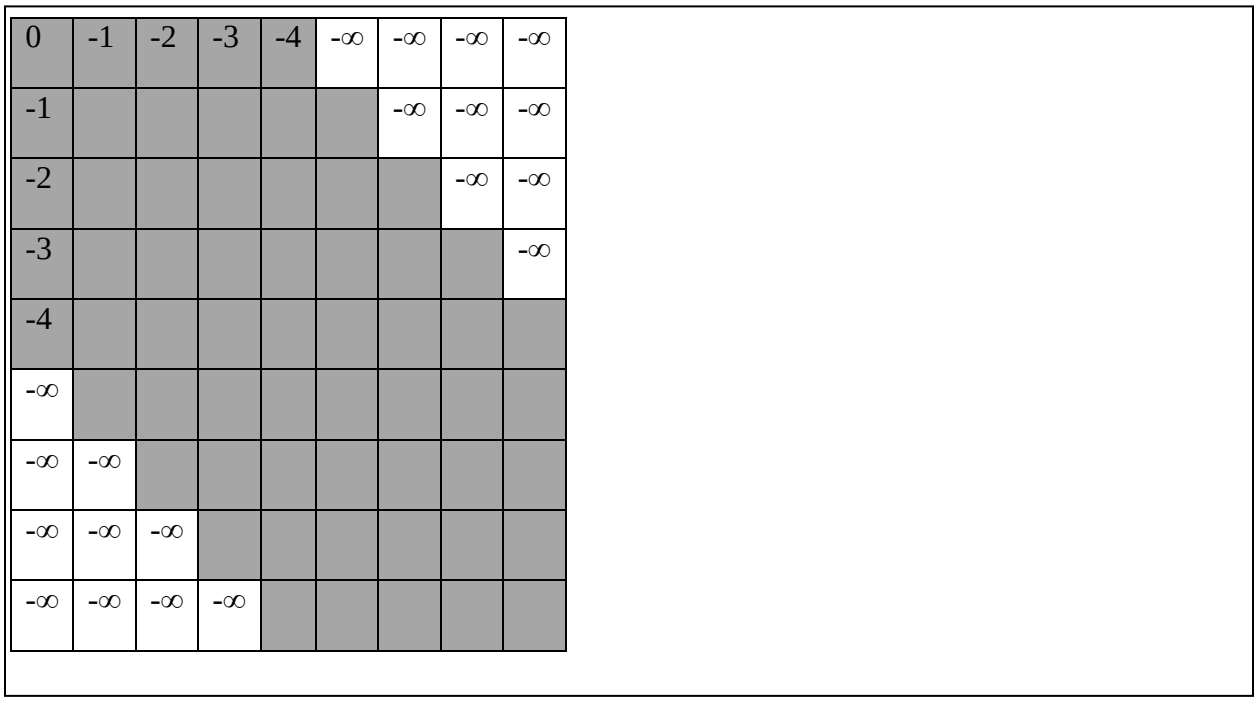

Figura 21. Ejemplo de celdas usadas por el algoritmo de K-Band

Como se puede notar en la figura el algoritmo lo que hace es ignorar todo lo que no se encuentre dentro de las bandas de tamaño *K*. Podemos saber si una celda está dentro de las bandas con la siguiente fórmula (Jackson & Aluru, 2001):

$$S[i,j] \in KBand \leftrightarrow |i-j| \leq k$$

Fórmula 2 – Celda dentro de las bandas

En esta fórmula, si el valor de *i* y *j* no es menor o igual que *K* se considera fuera de la banda, por lo que se coloca un valor de -∞. El algoritmo es igual al de Needleman-Wunsch, con la única condición de que si la celda se encuentra fuera de la banda no se va a usar para cálculos. Al final del algoritmo, vamos a conseguir un alineamiento óptimo para la matriz generada con los términos de *K* (Jackson & Aluru, 2001).

2.3.1.1 Alineamiento Óptimo mediante K-Band

Existen varios métodos para conseguir el alineamiento óptimo mediante el uso de K-Band. Varios de ellos implican repetir el algoritmo varias veces con el fin de encontrar el resultado óptimo en algún momento. Un listado de algunos de estos métodos son los siguientes:

1. **Fuerza bruta:** este método conlleva ejecutar el algoritmo una cantidad *X* de veces. Se termina la ejecución del mismo una vez que los resultados de la corrida *X-1 = X* (Huson, 2003).
2. **Eliminar K-Band:** colocar un valor de *K* con un número lo suficientemente grande para que abra toda la matriz y evitar el uso de una parte de la matriz (Huson, 2003).
3. **Valor de alineación:** mediante una ecuación al finalizar de ejecutar el algoritmo, se puede validar si el alineamiento conseguido es óptimo o no; si lo fuera se deja de correr el algoritmo, en caso contrario se incrementa el valor de *K* y se vuelve a correr. La ecuación se ejemplifica de la siguiente forma: se tienen dos secuencias *X* y *Y* ambas del mismo tamaño *N*, se tiene un valor de alineación *M*, un valor de gap penalti *d*, el mejor valor de la alineación *alphak* y la mejor alineación como *alpha*. Entonces se debe validar lo siguiente (Huson, 2003):

$$Si\ alphaK\ \geq M(n-k-1)-2(k+1)d\ entonces\ alphaK = alpha$$

Fórmula 3 – Validación del alineamiento obtenido usando K-Band

Lo que se quiere demostrar es que se va a conseguir un alineamiento óptimo cuando se tenga un valor de alineación que no se salga de la banda escogida. De lo contrario, el alineamiento óptimo se encuentra fuera de la banda, por lo que se necesita de una inserción de *k+1* ó *2k* gaps, dependiendo de lo que se ande buscando (Huson, 2003).

2.3.2 Ventajas y Desventajas del algoritmo

Este algoritmo tiene una utilidad alta en secuencias que se conocen parecidas; ya que así se puede estar casi seguros que con un *K* pequeño se puede tener el alineamiento óptimo.

En caso contrario puede ser importante también; ya que se puede estar buscando si dos secuencias no son iguales, como fue mencionado anteriormente, en el caso de las ciencias forenses es importante saber de forma rápida si dos secuencias de ADN son iguales o no, así poder ver quién cometió un delito. Es decir, si se encuentra una cantidad *X* de gaps se sabe que no es el individuo que se está buscando.

Una de las desventajas más grandes que se puede tener con este algoritmo es que puede ocasionar muchos problemas si lo que se anda buscando es un alineamiento óptimo, ya que si el *K* es más pequeño que la ruta óptima se va a tener un alineamiento óptimo para la banda *K* pero no sería la alineación óptima para ambas secuencias. Con esto, se puede decir que este algoritmo es rápido siempre y cuando consigamos el *K* correcto, en caso contrario estaríamos repitiendo el algoritmo (para *K = K+1* ó *2K*) hasta conseguir el *K* correcto (Jackson & Aluru, 2001).

Capitulo 3 Solución implementada

La solución propuesta consiste en una variante al algoritmo de K-Band. El mismo consigue una solución óptima recorriendo la matriz de similitud una única vez, a diferencia de K-Band que se debe de recorrer una cantidad *X* de tiempo. La principal característica de esta función es el abrir la matriz de similitud de la misma manera que K-Band pero en ciertos momentos se debe de abrir partes de la matriz que se encuentran fuera de las bandas; de esta forma se evita repetir el algoritmo más de una vez para conseguir un alineamiento óptimo.

Dada la complejidad del algoritmo implementado se explicará paso a paso las partes más importantes del mismo en las próximas secciones.

3.1 Variables usadas

En la solución implementada existen varios tipos de variables que son usadas alrededor de todo el algoritmo. Las más importantes se pueden encontrar a continuación:

1. **Done:** es usado para señalar el final de la alineación de las secuencias.
2. **Diag:** representa que el próximo valor a tomar de las matrices se encuentra en la diagonal superior izquierdo
3. **Left:** representa que el próximo valor a tomar de las matrices se encuentra en la celda izquierda
4. **Up:** representa que el próximo valor a tomar de las matrices se encuentra en la celda superior
5. **None:** representa que no hay valor en esa celda; es usado para las celdas que están fuera de las bandas.
6. **Alineamiento:** respuesta del alineamiento de las dos secuencias
7. **MS:** matriz de similitud de las secuencias
8. **MT:** matriz de traceback de las secuencias

3.2 Paso a paso

3.2.1 Revisión de los K escogidos

En el capítulo 2.3 se menciona que en K-Band se puede tener una banda de valor *K* para delimitar las dos principales diagonales *X* y *Y* en la matriz. Sin embargo, se puede tener valores de *K* distintos para cada una de las diagonales obteniendo: *K_X* y *K_Y*. Esto es lo mismo que tener dos bandas que limiten la apertura de la matriz. La solución implementada hace uso de esta metodología de dos bandas para poder así abrir la matriz de forma necesaria evitando el gasto de memoria o disco.

Si se hace uso del sistemas de bandas con un único valor de *K* podemos tener problemas al tratar de encontrar la alineación de dichas secuencias; esto pasa porque se va a tratar de obtener la alineación siempre empezando por el punto *NxM,* ya que se está usando alineamientos globales, pero dicho punto no tendrá aun valores. Tomando en cuenta lo antes mencionado, se puede ejemplificar de la siguiente manera:

Se tienen dos secuencias, una de tamaño cuatro, la otra de tamaño siete y se va a utilizar un K de uno.

		-∞	-∞
			-∞
-∞			
-∞	-∞		
-∞	-∞	-∞	
-∞	-∞	-∞	-∞
-∞	-∞	-∞	-∞

Figura 22. Ejemplo de manipulación de las bandas por el algoritmo implementado

Al observar detenidamente el ejemplo anterior, podemos notar que aún no es posible conseguir una respuesta para el alineamiento ya que la posición *N, M* no está inicializada. Este problema se puede atacar de varias maneras y a continuación se mencionan dos:

1. Una vez que se terminan de llenar las matrices (similitud y traceback) validar si la posición *N, M* se encuentra inicializada; de no ser así, aumentar el valor de *K*.
2. Utilizar la siguiente fórmula para conseguir los valores de *K* correctos para cada una de las bandas.

$$Si\ col > row\ \rightarrow kRow = col - row$$
$$Si\ row > col\ \rightarrow kCol = row - col$$

Fórmula 4 – Manipulación de las bandas para crear la matriz

La fórmula propuesta lo que realiza es cambiar el valor de *K* para cada una de las bandas que lo necesiten mediante los siguientes pasos:

1. Encontrar cual secuencia tiene mayor cantidad de caracteres. Si es la secuencia uno (se estaría hablando de col) o si es la secuencia dos (se hablaría de row).
2. Validar si la resta entre las cantidades de caracteres es mayor que el valor de *K* que se quiere cambiar en este caso. De ser así, el nuevo valor de *K* seria la resta de las cantidades de caracteres, en caso contrario no se realizaría ningún cambio porque la banda elegida es correcta.

Usemos el ejemplo anterior como demostración a esta fórmula:

- row es mayor que _col ya que row es 7 y col es 4
- row - col = 7 – 4 = 3
- 3 > krow = 3 > 1
- Como la resta entre las cantidades de las secuencias es mayor que el krow, cambiamos el valor por dicha resta.

Una vez que se cambian los valores de *K* podemos obtener la siguiente matriz.

		-∞	-∞
			-∞
-∞			
-∞	-∞		
-∞	-∞	-∞	

Figura 23. Ejemplo de las bandas manipuladas por el algoritmo implementado

De esta forma ya podemos obtener un resultado para la alineación de dichas secuencias; ya que la celda *N*, *M* se encuentra inicializada.

3.2.2 Inicializar la Matriz

Este paso es el mismo utilizado en el algoritmo de Needleman-Wunsch; lo que se hace es inicializar la matriz únicamente en los bordes superior e izquierdo de las matrices. La única diferencia es que se respetan las bandas elegidas.

3.2.3 Crear matriz de similitud y traceback

Este paso al mismo que el anterior es el mismo usado por Needleman-Wunsch. Este paso realiza dos acciones que son de suma importancia, las cuales son:

1) Colocar el valor de la celda *i, j* de la matriz de similitud
2) Colocar el valor de la celda *i, j* de la matriz de traceback

Para que este paso pueda ser usado por K-Band o cualquier variante derivada del mismo, estos pasos deben modificarse ligeramente para que pueda funcionar con el sistema de bandas. Se deben de validar los siguientes pasos:

1) Validar que la celda *i, j* se encuentra dentro de las bandas; de no ser así no se debe de calcular
2) Validar que los valores de las celdas $i - 1$ ó $j - 1$ se encuentran dentro de las bandas; de no ser asi esas celdas serán omitidas para el cálculo de la celda *i, j*.

Este algoritmo lo que realiza es que llena las matrices desde $i = 1$ hasta N y desde $j=1$ hasta M. El cálculo de la celda *i, j* se consigue tomando el valor máximo entre los campos: superior ($i - 1$, j), superior izquierdo ($i - 1$, $j - 1$) e izquierdo (i, $j - 1$) y en la matriz de traceback se coloca un puntero con la dirección de la opción tomada.

0	-4	-8	-∞	-∞	-∞	-∞
-4	1	-3	-7	-∞	-∞	-∞
-8	-3	2	-2		-∞	-∞
-∞	-7	-2	3			-∞
-∞	-∞	-6	-1			
-∞	-∞	-∞	-5			
-∞	-∞	-∞	-∞			

Figura 24. Ejemplo de creación de celdas por K-Band

Si se toma como ejemplo la matriz anterior, tomando como referencia el punto 2,3 podemos observar que el valor superior es de -7, el superior izquierdo de -3 y el izquierdo de 2; el algoritmo tomaría el valor de la izquierda y lo colocaría en la celda *i, j* de la matriz de similitud y en la matriz de traceback colocaría un valor que represente el lado izquierdo (con el fin de simbolizar de donde tomo dicho valor). Al mismo tiempo, se puede ejemplificar lo que pasa cuando la posición *i, j* está en el borde de la banda; si se toma de ejemplo la celda 1,3, en la misma se ignora la celda superior ya que se encuentra fuera de la banda y se utiliza únicamente los valores de -8 (superior izquierdo) y -3 (izquierdo) tomando el -3 como el valor máximo y colocando en el traceback el valor que representa la izquierda.

3.2.4 Obtener el resultado

Una vez que se termina de llenar la matriz, para Needleman-Wunsch y K-Band, lo único que hace falta es conseguir el resultado final. Este resultado se consigue fácilmente con la matriz de traceback; lo único que se debe de hacer es seguir el camino señalado en el traceback. Siempre se va a empezar en la posición *N,M,* y a partir de esa celda se empieza a conseguir el resultado siguiendo los siguientes pasos:

1) La celda tiene un valor que represente la dirección de arriba; por lo que la siguiente celda a validar seria *N – 1, M*.
2) La celda tiene un valor que represente la dirección de izquieda; por lo que la siguiente celda a validar seria *N, M – 1*.
3) La celda tiene un valor que represente la dirección de diagonal superior izquierda; por lo que la siguiente celda a validar seria *N - 1, M – 1*.

Una vez que se logre llegar a la posición *0, 0* se puede afirmar que se cuenta con una respuesta.

Como se ha mencionado, K-Band va a conseguir un resultado óptimo para la cantidad de bandas que se eligieron, pero eso no quiere decir que va a conseguir el resultado óptimo de las secuencias a alinear. Por lo antes mencionado, se implementó un cambio que valida si la posición *I, J* se encuentra en el límite de la banda y realiza los siguientes pasos:

1) Si no se encuentra en el límite de la banda, consiga el siguiente valor en el traceback.
2) Si se encuentra en el límite de la banda, se debe expandir el valor de la banda hasta la posición *I - 1* o *J - 1*. Con el fin de evitar la incertidumbre que genera un valor el límite de la banda; el estar en el límite de la banda implica que se ignoró alguna de las celdas (superior, superior izquierda, izquierda) pudiendo ser esta la próxima respuesta para un alineamiento óptimo.

Seguidamente, se realizará un ejemplo de esta nueva característica y se contrastará contra Needleman-Wunsch y K-Band para que se pueda entender gráficamente. Se toma la secuencia 1 como HEAGAWGHEE y la secuencia 2 como PAWHEAE.

```
Function TraceBack(matrix)
{
   I = count(sequence1)
   J = count(sequence2)
   Path = matrix[I,J]
   While(Path <> matriz[0,0])
   {
     inBorder = IsInBorder(I,J)
     switch(inBorder)
     {
       Case 'LEFT':
         OpenRowColumnOnMatrix(inBorder,I,J+1)
       Case 'UP':
         OpenRowColumnOnMatrix(inBorder,I+1,J)
     }
     Answer = Path + Answer
     If (Path == 'DIAG')
       I--
       J--
     If (Path == 'UP')
       I--
     ELSE
       J--
   }
}
Function OpenRowColumnOnMatrix(type,I,J)
{
    Minor = Min(I,J)
    auxI = I - minor
    auxJ = J - minor
    Initilize(auxI,auxJ)
    auxI++
    auxJ++
    while(I >= auxI AND J >= auxJ)
    {
      GetResults(auxI,aux)
      If(type = 'LEFT')
        auxJ++
      If(type = 'UP')
         auxI++
}
```

Figura 25. Pseudocódigo de la mejora planteada

3.2.4.1 Needleman-Wunsch

La matriz de traceback del algoritmo de Needleman-Wunsch se ve de la siguiente manera.

		P	A	W	H	E	A	E
	*	-	-	-	-	-	-	-
H	|	\	\	\	\	-	-	-
E	|	\	\	\	|	\	-	\
A	|	\	\	-	-	|	\	-
G	|	\	|	\	\	|	|	\
A	|	\	\	\	\	\	\	\
W	|	\	|	\	-	\	|	\
G	|	\	|	|	\	\	\	\
H	|	\	|	|	\	-	-	\
E	|	\	|	|	|	\	-	\
E	|	\	|	|	|	\	\	\

Figura 26. Matriz de traceback de Needleman-Wunsch.
El color anaranjado denota el camino a tomar para obtener la respuesta.

3.2.4.2 K-Band

La matriz de traceback del algoritmo de K-Band se ve de la siguiente manera.

Las bandas tendrán el mismo valor por lo que se utilizara un valor de $K = 3$.

		P	A	W	H	E	A	E
	*	-	-	-	#	#	#	#
H	|	\	\	\	\	#	#	#
E	|	\	\	\	|	\	#	#
A	|	\	\	-	-	|	\	#
G	#	\	|	\	\	|	|	\
A	#	#	\	\	\	\	\	\
W	#	#	#	\	-	\	|	\
G	#	#	#	#	\	\	\	\
H	#	#	#	#	#	\	\	\
E	#	#	#	#	#	#	\	\
E	#	#	#	#	#	#	#	\

Figura 27. Matriz de traceback de K-Band.
El color anaranjado denota el camino a tomar para obtener la respuesta, y el color gris denota las celdas sin usar.

3.2.4.3 Mejora

Las bandas tendrán el mismo valor por lo que se utilizara un valor de $K = 3$.

Primeramente, se genera la matriz de traceback de la misma manera que K-Band. A la hora de conseguir la respuesta se va a ir preguntando si estamos en el borde o no. Se inicia en la celda *N*, *M*.

		P	A	W	H	E	A	E
	*	-	-	-	#	#	#	#
H	|	\	\	\	\	#	#	#
E	|	\	\	\	|	\	#	#
A	|	\	\	-	-	|	\	#
G	#	\	|	\	\	|	|	\
A	#	#	\	\	\	\	\	\
W	#	#	#	\	-	\	|	\
G	#	#	#	#	\	\	\	\
H	#	#	#	#	#	\	\	\
E	#	#	#	#	#	#	\	\
E	#	#	#	#	#	#	#	\

Figura 28. Matriz de traceback del algoritmo creado 1 de 5.
El color anaranjado denota el camino a tomar para obtener la respuesta, y el color gris denota las celdas sin usar.

Como esa celda *E,E* se encuentra en el borde se va a proceder a expandir la banda inferior.

		P	A	W	H	E	A	E
	*	-	-	-	#	#	#	#
H	|	\	\	\	\	#	#	#
E	|	\	\	\	|	\	#	#
A	|	\	\	-	-	|	\	#
G	|	\	|	\	\	|	|	\
A	#	\	\	\	\	\	\	\
W	#	#	|	\	-	\	|	\
G	#	#	#	|	\	\	\	\
H	#	#	#	#	\	-	-	\
E	#	#	#	#	#	\	-	\
E	#	#	#	#	#	#	\	\

Figura 29. Matriz de traceback del algoritmo creado 2 de 5.
El color anaranjado denota el camino a tomar para obtener la respuesta, el color gris denota las celdas sin usar y el color morado denota el cambio en las celdas de la matriz.

Una vez que se expande la banda hay que tomar en cuenta que pueden cambiar celdas cercanas a ese borde por lo que hay que hacer una validación de si la celda más próxima al borde cambio o no, y así sucesivamente para verificar todos los cambios que puede llegar a sufrir la matriz. Por ejemplo la celda *E, H* cambio porque *H, H* cambio y gracias a que esta cambio *A, H* también llego a cambiar porque los valores de la izquierda cambiaron; sin embargo, *E, H* no cambio del todo por lo que no debimos cambiar nada en él.

Si se sigue con el recorrido de la matriz encontramos que la posición *E,E* se encuentra en el borde y debemos nuevamente expandir la banda inferior.

		P	A	W	H	E	A	E
	*	-	-	-	#	#	#	#
H	|	\	\	\	\	#	#	#
E	|	\	\	\	|	\	#	#
A	|	\	\	-	-	|	\	#
G	|	\	|	\	\	|	|	\
A	#	\	\	\	\	\	\	\
W	#	#	|	\	-	\	|	\
G	#	#	#	|	\	\	\	\
H	#	#	#	#	\	-	-	\
E	#	#	#	#	#	\	-	\
E	#	#	#	#	#	#	\	\

Figura 30. Matriz de traceback del algoritmo creado 3 de 5.
El color anaranjado denota el camino a tomar para obtener la respuesta, y el color gris denota las celdas sin usar.

Se expande el valor de la banda inferior tomando en cuenta que no se tiene que abrir toda la banda; sino abrir hasta la casilla *H, E*. Ya que los valores que se encuentran abajo no van a ser usados y no se quiere gastar memoria o disco con celdas innecesarias.

		P	A	W	H	E	A	E
	*	-	-	-	#	#	#	#
H	|	\	\	\	\	#	#	#
E	|	\	\	\	|	\	#	#
A	|	\	\	-	-	|	\	#
G	|	\	|	\	\	|	|	\
A	|	\	\	\	\	\	\	\
W	#	\	|	\	-	\	|	\
G	#	#	|	|	\	\	\	\
H	#	#	#	|	\	-	-	\
E	#	#	#	#	|	\	-	\
E	#	#	#	#	#	#	\	\

Figura 31. Matriz de traceback del algoritmo creado 2 de 5.
El color anaranjado denota el camino a tomar para obtener la respuesta, el color gris denota las celdas sin usar y el color morado denota el cambio en las celdas de la matriz.

Una vez concluido este paso se tiene la alineación óptima de las dos secuencias suministradas.

		P	A	W	H	E	A	E
	*	-	-	-	#	#	#	#
H	|	\	\	\	\	#	#	#
E	|	\	\	\	|	\	#	#
A	|	\	\	-	-	|	\	#
G	|	\	|	\	\	|	|	\
A	|	\	\	\	\	\	\	\
W	#	\	|	\	-	\	|	\
G	#	#	|	|	\	\	\	\
H	#	#	#	|	\	-	-	\
E	#	#	#	#	|	\	-	\
E	#	#	#	#	#	#	\	\

Figura 32. Matriz de traceback del algoritmo creado 3 de 5.
El color anaranjado denota el camino a tomar para obtener la respuesta, y el color gris denota las celdas sin usar.

Con este ejemplo se puede ver el cambio que se tiene entre los 3 algoritmos y como la mejora consigue la misma secuencia optima que Needleman.

Con la modificación creada al algoritmo de K-Band se puede asegurar una respuesta óptima siempre, al igual que al utilizar Needleman-Wunsch. Esto debido a que no se da campo a la incertidumbre que se genera cuando el algoritmo se encuentra en una celda que está en el borde de alguna de las bandas; esto se evita aumentando la banda para conseguir el resultado. Al mismo tiempo, se puede asegurar que vamos a poder utilizar más eficientemente la memoria o el disco ya que las celdas que no son requeridas se ignoran. Cabe resaltar que para casos de búsqueda de alineamientos óptimos, este nuevo algoritmo es mucho más eficiente que K-Band ya que no tiene recrear toda la matriz una y otra vez hasta encontrar la respuesta; sólo debe crearse una vez.

Capitulo 4 Experimentos

Con los datos ya estructurados y seleccionados, puede procederse con múltiples experimentos para comparar el desempeño de la variante del algoritmo contra Needleman-Wunsch y K-Band. La variante del experimento consiste en la elección correcta de las dos secuencias y ejecutar el algoritmo sobre las secuencias para obtener las alineaciones de cada uno.

Los experimentos a realizar cambiaran únicamente en las secuencias a usar. A partir de las secuencias elegidas o creadas, se realizarán dos tipos de experimentos:

1) Validación puntual sobre dos secuencias con el fin de determinar la eficiencia del algoritmo en cuanto a tiempo, espacio en disco o memoria.
2) Creación de un Benchmark con el que se podrá tomar los tiempos de duración de cada uno de los algoritmos sobre dos secuencias creadas a partir de un universo *U* y repetir las alineaciones una cantidad *X* de veces.

Los resultados a tomar para cada experimento serán los siguientes:

- Alineamiento de las secuencias dadas junto con la validación de la matriz de similitud y traceback. Esto permite validar que los algoritmos llegan a una respuesta óptima a partir de los valores dados. Si los resultados son iguales que Needleman-Wunsch se puede decir que la alineación es la correcta.
- Tiempo total de ejecución; será tomado a partir de que el algoritmo empieza a correr hasta que se logra validar la veracidad de la respuesta. Toda operación de tipo Entrada o Salida no es tomada en cuenta. En este caso se utilizarán los *Ticks* del reloj para validar el tiempo de ejecución ya que los algoritmos corren muy rápido en secuencias pequeñas.
- Espacio en disco / memoria usado; se medirá con la ayuda de la cantidad de celdas usadas al final por la matriz.

4.1 Requisitos del sistema

Los experimentos realizados fueron ejecutados en una computadora con las siguientes características:

1) Sistema Operativo: Windows 7 Enterprise 64 bits - Service Pack 1
2) RAM: 16 GB
3) Procesador: Intel® Core™ i7-3520M CPU @ 2.90GHz
4) Disco duro: 225 GB – Intel SSD

4.2 Datos de prueba

Este algoritmo toma como entrada dos secuencias de elementos de un universo U, los cuales equivalen a caracteres del alfabeto. Si se realiza el Benchmark el mismo va a tomar un número que equivale a la cantidad de veces que se correrá el algoritmo sobre las secuencias.

Por lo tanto, se creó un generador de secuencias. Genera secuencias de tamaños *N* y *M* del universo escrito. Estas secuencias serán usadas por los algoritmos para poder contrastar el tiempo de ejecución de los mismos.

4.3 Experimentos concretos

El conjunto de pruebas aquí descrito está destinado a proveer un panorama general de la eficacia del algoritmo. Se van a tomar distintas secuencias y variando el *K* suministrado para ver el desempeño de los mismos.

La lista de los experimentos ejecutados es la siguiente:

Experimento	**Descripción**
1	Tomar secuencias conocidas con un *K = 2*
2	Tomar la misma secuencia que en el experimento 1 con un *K=4*
3	Tomar secuencias idénticas con un *K = 2*
4	Benchmark#1 repetir los pasos del experimento 1 una cantidad de 1.000 veces por algoritmo
5	Benchmark#2 repetir la alineación con secuencias creadas por el algoritmo *Random Sequence* con tamaños *N = 150* y *M = 200* una cantidad de 1.000 veces para los tres algoritmos
6	Benchmark#3 repetir la alineación con secuencias creadas por el algoritmo *Random Sequence* con tamaños *N = 100* y *M = 200* una cantidad de 1.000 veces para los tres algoritmos; con el alineamiento próximo al centro, izquierda y arriba de la matriz

Tabla 1 – Lista de experimentos

4.3.1 Experimento 1

Descripción: Tomar secuencias conocidas con un K = 2

- Secuencia 1: HEAGAWGHEEAAA
- Secuencia 2: PAWHEAEAAA
- K: 2

	Needleman-Wunsch	K-Band	Mejora
Secuencia1	HEAGAWGHE-EAAA	HEAGAWGHE-EAAA	HEAGAWGHE-EAAA
Secuencia2	---PAW-HEAEAAA	---PAW-HEAEAAA	---PAW-HEAEAAA
Ticks	64	144	74
Celdas Usadas	154	97	78
Corridas	1	3	1

Tabla 2 – Resultados de experimento 1

4.3.2 Experimento 2

Descripción: Tomar la misma secuencia que en el experimento 1 con un K=4

- Secuencia 1: HEAGAWGHEEAAA
- Secuencia 2: PAWHEAEAAA
- K: 4

	Needleman-Wunsch	K-Band	Mejora
Secuencia1	HEAGAWGHE-EAAA	HEAGAWGHE-EAAA	HEAGAWGHE-EAAA
Secuencia2	---PAW-HEAEAAA	---PAW-HEAEAAA	---PAW-HEAEAAA
Ticks	65	105	62
Celdas Usadas	154	103	93
Corridas	1	2	1

Tabla 3 – Resultados de experimento 2

4.3.3 Experimento 3

Descripción: Tomar secuencias idénticas con un K = 2

- Secuencia 1: HEAGAWGHEEAAA
- Secuencia 2: HEAGAWGHEEAAA
- K: 2

	Needleman-Wunsch	K-Band	Mejora
Secuencia1	HEAGAWGHEEAAA	HEAGAWGHEEAAA	HEAGAWGHEEAAA
Secuencia2	HEAGAWGHEEAAA	HEAGAWGHEEAAA	HEAGAWGHEEAAA
Ticks	74	139	48
Celdas Usadas	196	86	64
Corridas	1	2	1

Tabla 4 – Resultados de experimento 3

4.3.4 Experimento 4

Descripción: Benchmark#1 repetir los pasos del experimento 1 una cantidad de 1.000 veces por algoritmo

- Secuencia 1: HEAGAWGHEEAAA
- Secuencia 2: PAWHEAEAAA
- K: 2
- Cantidad de repeticiones: 1,000

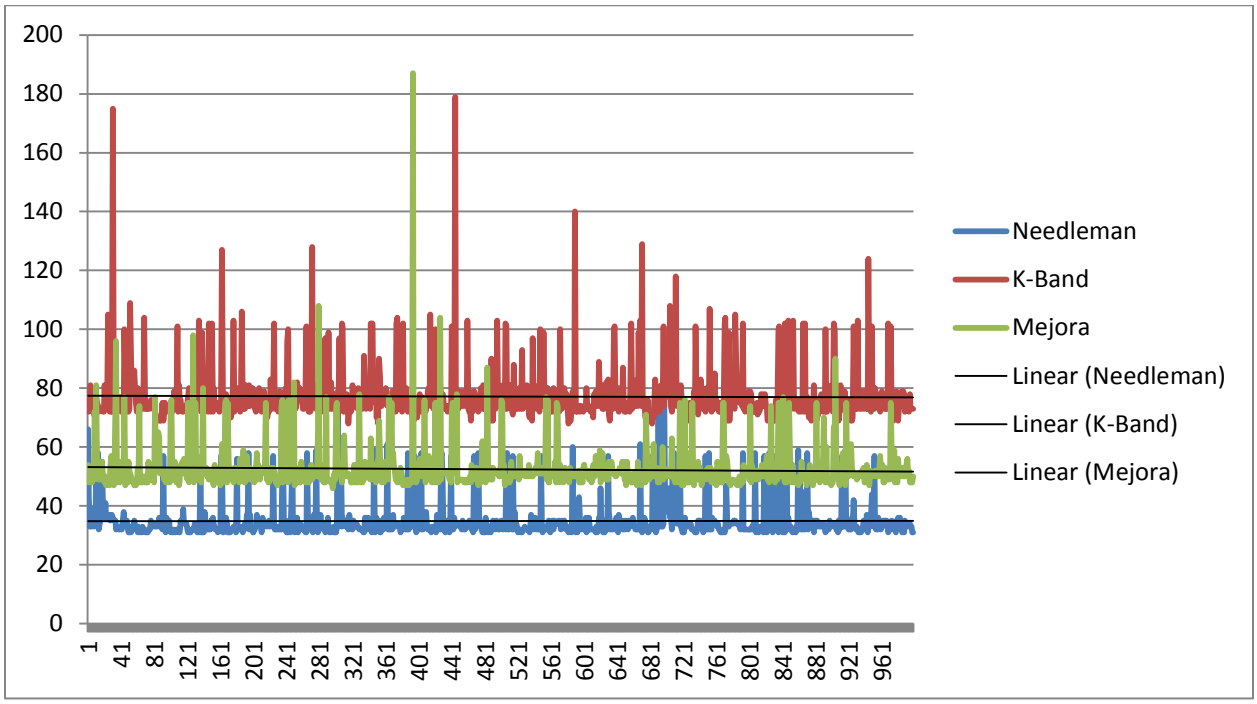

Figura 33. Resultados del experimento 4
En esta figura se puede encontrar la cantidad de ticks que duro cada una de las iteraciones del algoritmo para dos secuencias estipuladas.

	Needleman-Wunsch	**K-Band**	**Mejora**
Iteraciones	1,000	1,000	1,000
Total de Ticks	34,822	77,080	52,373
Media	34.8	77.1	52.4

Tabla 5 – Resultados de experimento 4

4.3.5 Experimento 5

Descripción: Benchmark#2 repetir la alineación con secuencias creadas por el random sequence con tamaños N = 200 y M = 150 una cantidad de 1.000 veces para los tres algoritmos

- Alfabeto: STCNQYGAVLIMPFWDEKRH
- Secuencia 1: random a partir del alfabeto con una cantidad de 200 caracteres
- Secuencia 2: random a partir del alfabeto con una cantidad de 150 caracteres
- K: 5
- Cantidad de repeticiones: 1,000

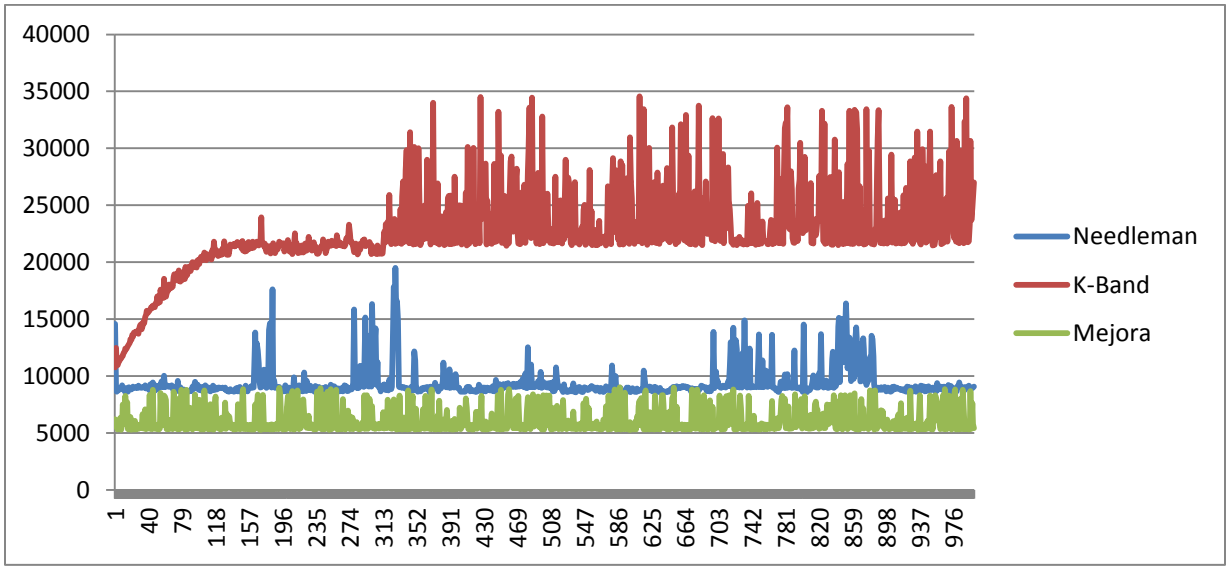

Figura 34. Resultados del experimento 5
En esta figura se puede encontrar la cantidad de ticks que duro cada una de las iteraciones del algoritmo para dos secuencias estipuladas.

	Needleman-Wunsch	**K-Band**	**Mejora**
Iteraciones	1,000	1,000	1,000
Total de Ticks	9,342,622	22,720,338	6,127,959
Media de Ticks	9,342.6	22,720.3	6,128
Total de Milisegundos	3,092	7,474	1,601
Media de Milisegundos	3.1	7.5	1.6

Tabla 6 – Resultados de experimento 5

4.3.6 Experimento 6

Descripción: *Benchmark#3 repetir la alineación con secuencias creadas por el algoritmo Random Sequence con tamaños N = 100 y M = 200 una cantidad de 1.000 veces para los tres algoritmos; con el alineamiento próximo al centro, izquierda y arriba de la matriz*

- Alfabeto: STCNQYGAVLIMPFWDEKRH
- Secuencia 1: random a partir del alfabeto con una cantidad de 100 caracteres
- Secuencia 2: random a partir del alfabeto con una cantidad de 200 caracteres
- K: 5
- Cantidad de repeticiones: 1,000

4.3.6.1 Alineamiento próximo al centro

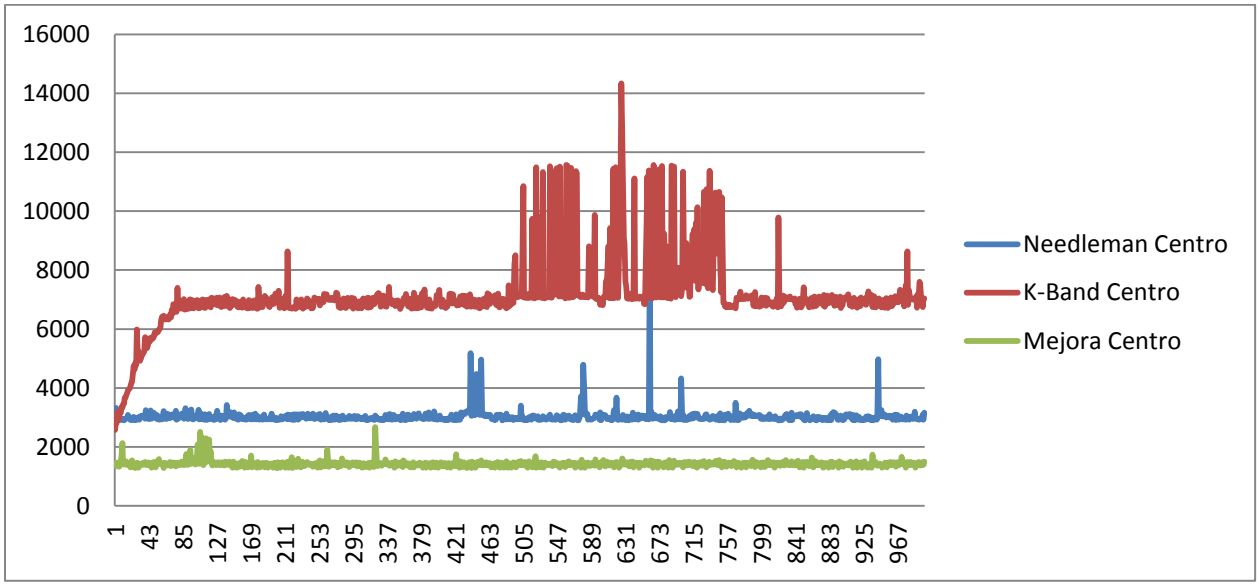

Figura 35. Resultados del experimento 6 con un alineamiento próximo al centro
En esta figura se puede encontrar la cantidad de ticks que duro cada una de las iteraciones del algoritmo para dos secuencias estipuladas.

	Needleman-Wunsch	K-Band	Mejora
Iteraciones	1,000	1,000	1,000
Total de Ticks	3,024,966	7,112,131	1,426,679
Media de Ticks	3,025	7,112,1	1,426,7
Total de Milisegundos	1,004	1,002	0
Media de Milisegundos	1	1	0

Tabla 7 – Resultados de experimento 6 con un alineamiento próximo al centro

4.3.6.2 Alineamiento próximo a la izquierda

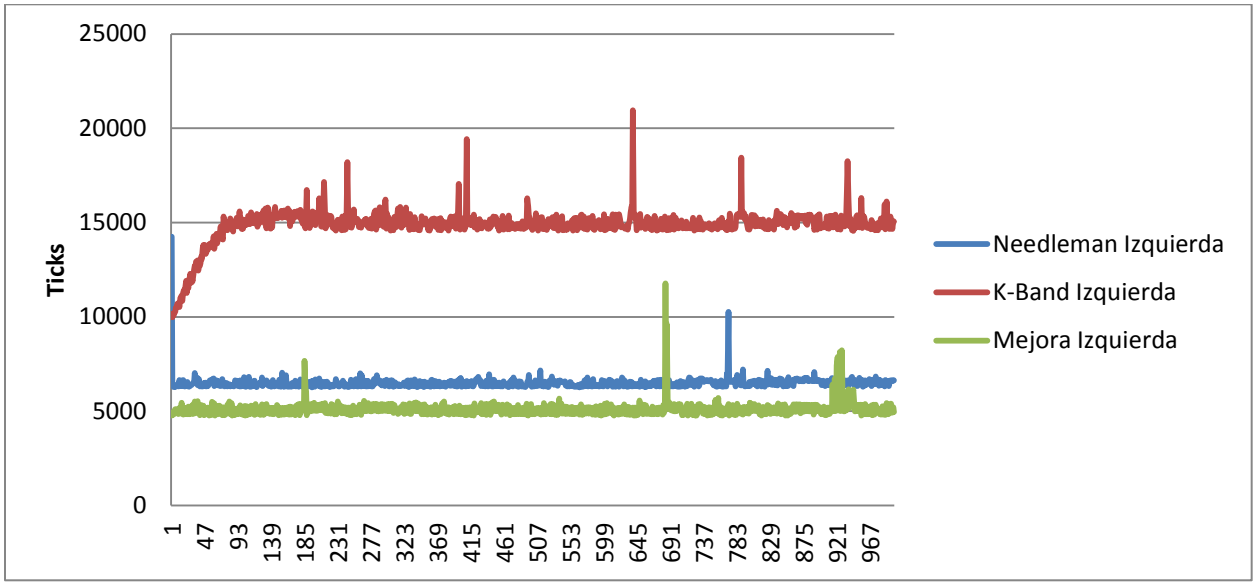

Figura 36. Resultados del experimento 6 con un alineamiento próximo a la izquierda
En esta figura se puede encontrar la cantidad de ticks que duro cada una de las iteraciones del algoritmo para dos secuencias estipuladas.

	Needleman-Wunsch	K-Band	Mejora
Iteraciones	1,000	1,000	1,000
Total de Ticks	6,483,987	14,857,864	5,087,014
Media de Ticks	6,484	14,858	5,087
Total de Milisegundos	2,004	4,927	1,022
Media de Milisegundos	2	4.9	1

Tabla 8 – Resultados de experimento 6 con un alineamiento próximo a la izquierda

4.3.6.3 Alineamiento próximo hacia arriba

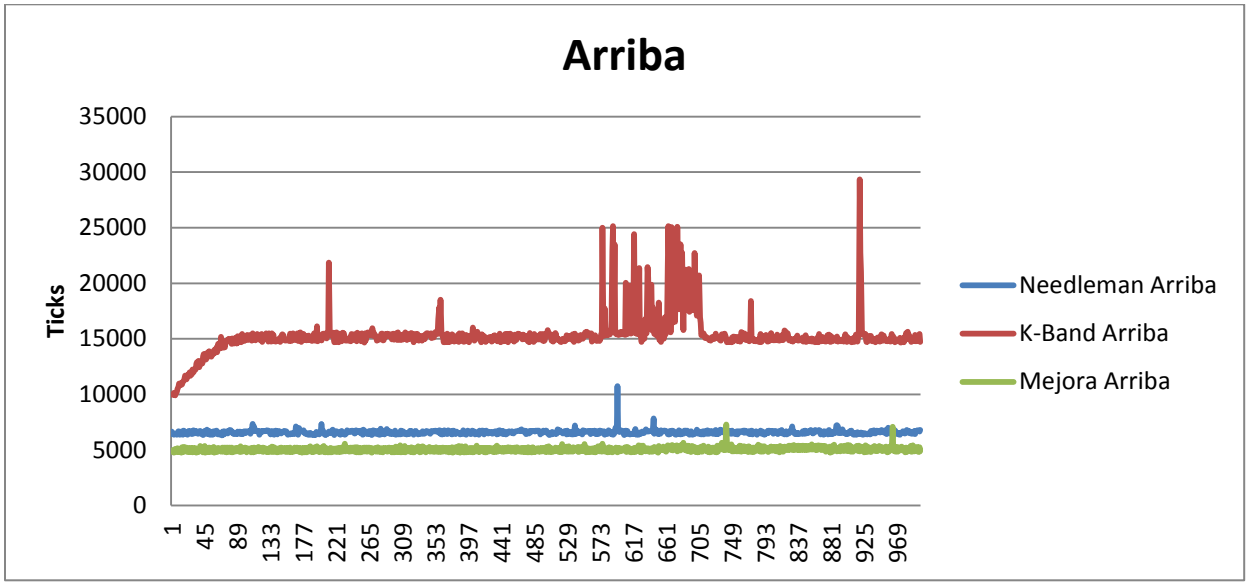

Figura 37. Resultados del experimento 6 con un alineamiento próximo hacia arriba
En esta figura se puede encontrar la cantidad de ticks que duro cada una de las iteraciones del algoritmo para dos secuencias estipuladas.

	Needleman-Wunsch	K-Band	Mejora
Iteraciones	1,000	1,000	1,000
Total de Ticks	6,589,011	15,284,966	5,026,932
Media de Ticks	6,589	15,285	5,026.9
Total de Milisegundos	2,001	5,039	1,004
Media de Milisegundos	2	5	1

Tabla 9 – Resultados de experimento 6 con un alineamiento próximo hacia arriba

4.3.6.4 Alineamiento basado en la mejora

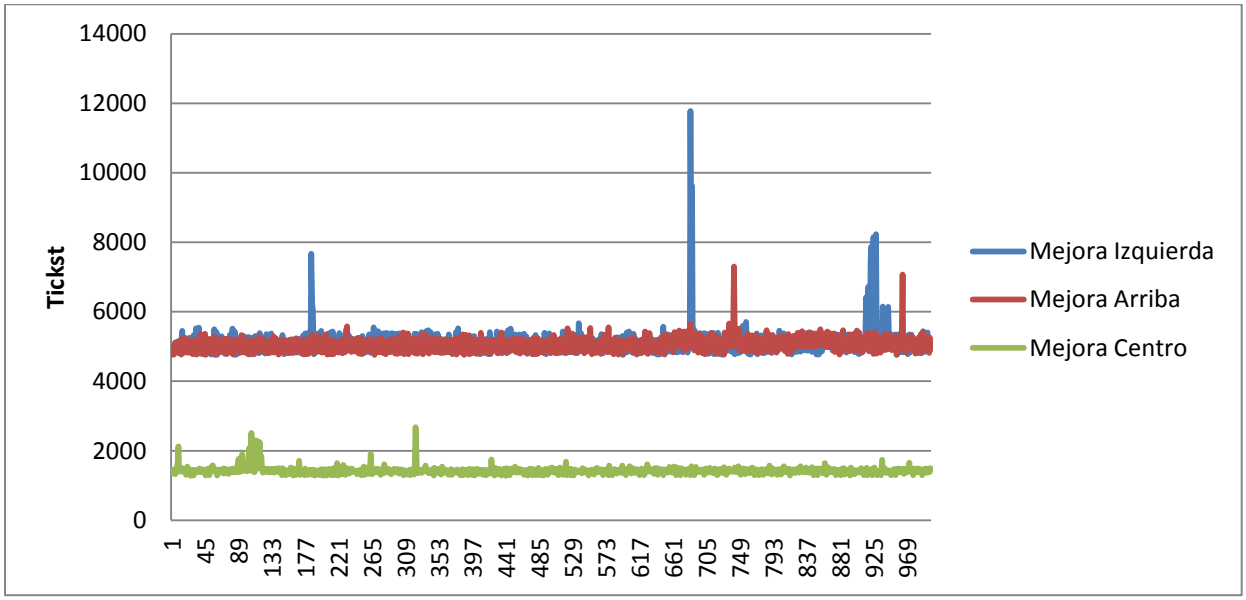

Figura 38. Resultados del experimento 6 basados en la mejora
En esta figura se puede encontrar la cantidad de ticks que duro cada una de las iteraciones del algoritmo para dos secuencias estipuladas.

	Mejora Izquierda	**Mejora Arriba**	**Mejora Centro**
Iteraciones	1,000	1,000	1,000
Total de Ticks	5,087,014	5,026,932	1,426,679
Media de Ticks	5,087	5,026.9	1,426,7
Total de Milisegundos	1,022	1,004	0
Media de Milisegundos	1	1	0

Tabla 10 – Resultados de experimento 6 basados en la mejora

Capitulo 5 Análisis de resultados

5.1 Comportamiento del manejo de la matriz

Se concluye a partir de los experimentos 1, 2 y 3 que el nuevo manejo de la matriz de similitud aumenta la disponibilidad de espacio en disco o memoria. Esto se puede notar con la disminución del uso de las matrices de similitud y traceback; dado que la mejora del algoritmo hace uso de una menor cantidad de celdas que el resto

5.2 Comportamiento con secuencias pequeñas

Se concluye a partir del experimento 4 que la mejora del algoritmo no es la mejor opción en cuanto a velocidad, ya que Needleman-Wunsch es más rápido; sin embargo la mejora sobrepasa a K-Band en casi la mitad del tiempo.

5.3 Comportamiento con secuencias grandes

Se concluye a partir del experimento 5 y 6 que la mejora del algoritmo es mucho más rápida computacionalmente que las implementaciones de Needleman-Wunsch y K-Band. Se logra demostrar que Needleman-Wunsch brinda una respuesta en el doble de tiempo que la mejora mientras que K-Band dura 5 veces más para dar la respuesta de las secuencias. Al mismo tiempo, se puede notar, gracias al experimento 6, que la mejora del algoritmo es mucho más eficiente cuando el alineamiento se encuentra en el centro de la matriz que cuando se encuentra en alguna otra parte.

5.4 Desempeño en milisegundos y ticks

Se concluye a partir del experimento 4, 5 y 6 que la mejora del algoritmo se comporta de una manera eficiente en cuanto a velocidad computacional, con secuencias de tamaño grande, en contraste con los otros algoritmos implementados. Al mismo tiempo, gracias a los experimentos realizados se puede probar que es mejor utilizar el algoritmo propuesto que K-Band, para secuencias pequeñas; ya que se pueden conseguir los resultados de una forma más rápida. Esto porque se utilizan más ticks del reloj para dar el resultado.

5.5 Análisis de ANOVA

Seguidamente se presenta los análisis de ANOVA para los experimentos 4 y 5. Los análisis se corrieron con el programa de Minitab 17. Ambos comparten la siguiente información:

- Hipótesis nula: Todas las medias son iguales
- Hipótesis alterna: Por lo menos una media es diferente
- Nivel de significancia $\alpha = 0.05$
- Varianzas iguales se asumen para el análisis

5.5.1 Experimento 4

5.5.1.1 Análisis de varianza

Source	**DF**	**Adj SS**	**Adj MS**	**F-Value**	**P-Value**
Factor	2	902,306	451,153	6,493.97	0.000
Error	2,994	208,001	69		
Total	2,996	1,110,307			

Tabla 11 – Análisis de varianza del experimento 4

5.5.1.2 Medias

Source	**N**	**Mean**	**St. Dev**	**95% CI**
Needleman	1,000	34.857	6.701	(34.340 , 35.374)
K-Band	1,000	77.157	9.651	(76.640 , 77.674)
Mejora	1,000	52.425	8.388	(51.908 , 52.942)

Tabla 12 – Análisis de medias del experimento 4

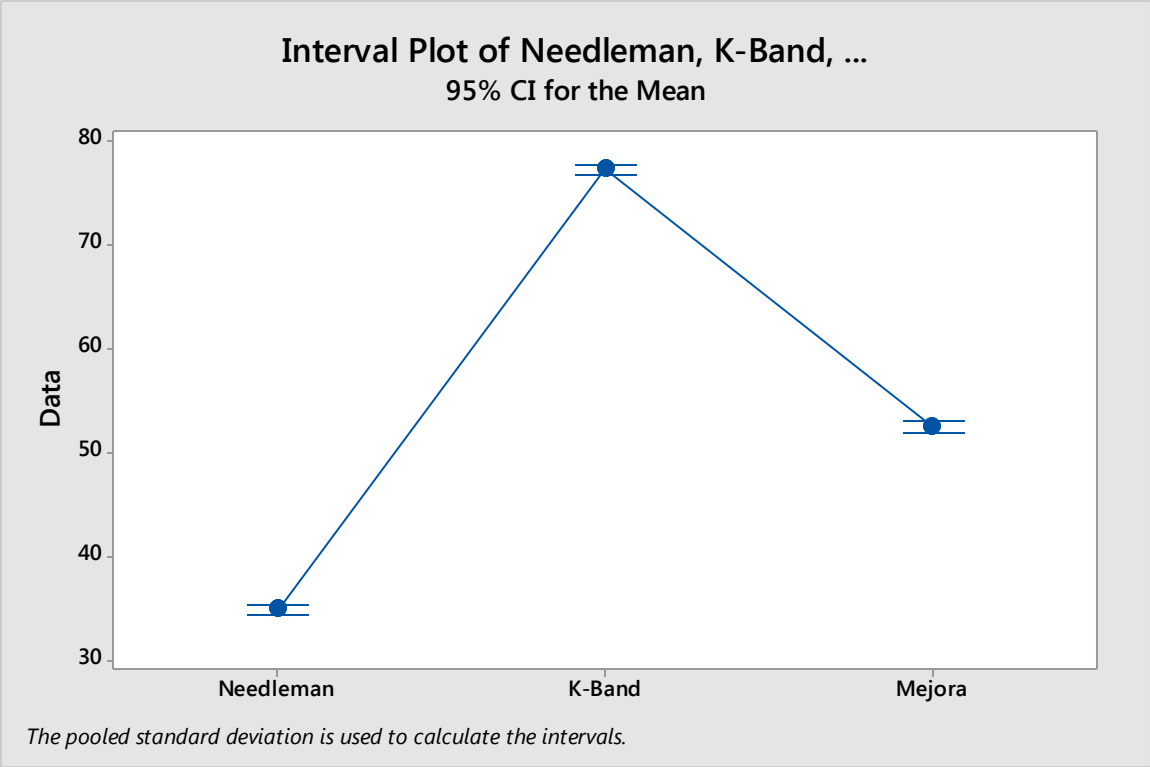

Figura 39. Gráfico de intervalos del experimento 4 usando ANOVA

5.5.2 Experimento 5

5.5.2.1 Análisis de varianza

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Factor	2	1.54868E+11	774,340,640,77	14,432.16	0.000
Error	2,997	16,080,051,242	5,365,382		
Total	2,999	1.70948E+11			

Tabla 13 – Análisis de varianza del experimento 5

5.5.2.2 Medias

Source	N	Mean	St. Dev	95% CI
Needleman	1,000	9,342.6	1,291.5	(9,199 , 9,486.2)
K-Band	1,000	22,720	3,655	(22,577 , 22,864)
Mejora	1,000	6,128	1,033.7	(5,984.3 , 6,271.6)

Tabla 14 – Análisis de medias del experimento 5

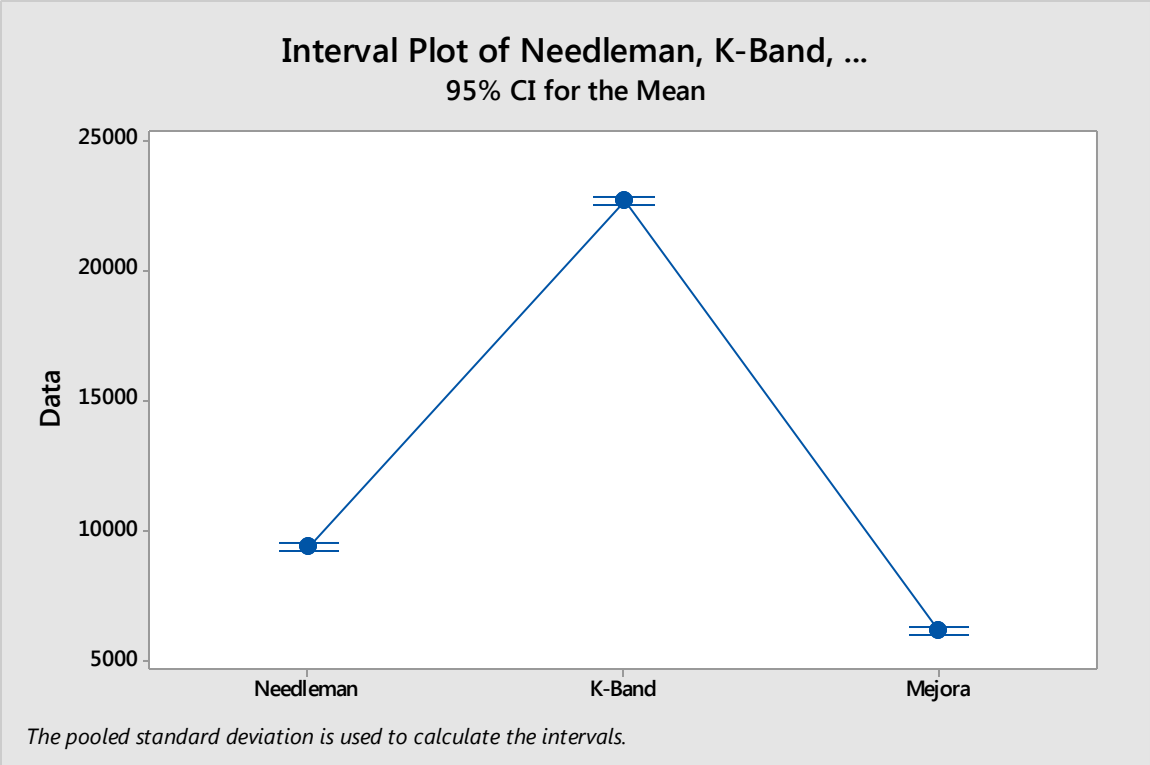

Figura 40. Gráfico de intervalos del experimento 5 usando ANOVA

Capitulo 6 Conclusiones

6.1 Resultado de los objetivos

El algoritmo descrito en el capítulo 3 satisface el objetivo 1: *Plantear y realizar ajustes al algoritmo para mejorar el rendimiento del mismo*. El pseudocódigo y el código fuente del algoritmo se encuentra adjunto a esta tesis para un futuro uso.

El objetivo 2, consiste en *Validar el algoritmo obtenido contra secuencias ya probadas en dicho algoritmo para confirmar la validez de las respuestas obtenidas por el algoritmo generado*. Fue llevado a cabo y comprobado en los resultados de los experimentos planteados en el capítulo 4. Cada respuesta dada por los algoritmos fue comparada, una contra otra, para validar que sean idénticas.

Finalmente, el objetivo 3 *Demostrar mediante un Benchmark que los ajustes realizados al algoritmo mejoran el rendimiento del mismo*. Fue llevado a cabo con los experimentos del 4 al 6. En los que se puede notar que se hace un menor uso del espacio en disco o memoria; al mismo tiempo, se reduce el tiempo computacional, con secuencias grandes, necesario para brindar una respuesta óptima. Si bien es cierto, Needleman-Wunsch es más rápido que la mejora planteada con secuencias pequeñas; sin embargo, los algoritmos de alineamiento se usan con secuencias de mucho mayor tamaño como es el caso del genoma humano con 3.000 millones de pares de bases, demostrando que las secuencias a alinear usadas son de un volumen mayor.

6.2 Veracidad de las Hipótesis

La hipótesis, *la mezcla de dos algoritmos de programación dinámica junto con la minimización del uso de la matriz de similitud es suficiente para reducir la complejidad espacial que presenta el algoritmo de Needleman-Wunsch* se comprobó mediante los siguientes puntos:

1) El algoritmo generado es una mezcla de Needleman-Wunsch y K-Band por lo que se unifican dos algoritmos de programación dinámica.
2) Los resultados de los experimentos planteados demuestran que se hace un menor uso de la cantidad de celdas de las matrices, por lo que estamos disminuyendo el uso de la matriz de similitud
3) El nuevo algoritmo es significativamente más eficiente a nivel de disco o memoria y tiempo de ejecución con respecto a Needleman-Wunsch demostrado con los resultados obtenidos de los experimentos.

4) El algoritmo propuesto utiliza ligeramente menor cantidad de celdas de la matriz de similitud que K-Band, sin embargo, es sustancialmente mejor en cuanto a tiempo de ejecución; por lo que el uso del mismo puede aumentar en casos que se necesite una respuesta óptima usando K-Band; dando la respuesta óptima en un menor tiempo de ejecución.
5) El algoritmo es de tipo optimista que sin cambios significativos en el análisis de $O(n)$ mejora significativamente el $O(n)$ tal y como se ven en los experimentos del capítulo 4.
6) El algoritmo funciona de mejor manera con secuencias que coinciden entre sí; lo que se adapta muy bien al análisis y comparación en ciertas clases de secuencias que cumplan esta propiedad
7) La mejora en la complejidad espacial se hace evidente cuanto más grandes son las hileras, lo que se evidencia en los resultados de los experimentos 5 y 6

6.3 Comentarios finales y futuras áreas de investigación

El área de Bioinformática ha ido creciendo poco a poco; a tal punto que, actualmente, es necesaria para poder investigar de forma eficiente (Transparency Market Research, 2012). Cada paso que se da en la ciencia, puede verse ayudado por algún mecanismo que facilite el trabajo. Por lo que es necesario de la constante mejora e invención de algoritmos que soporten la labor de los científicos de la época.

Los mecanismos de alineamiento de secuencias son bien conocidos y bastante estructurados. Por lo que se puede tomar como una solución final para este problema; sin embargo, se puede mejorar el rendimiento del mismo o el espacio o tiempo de ejecución para poder facilitar o agilizar el trabajo de personas.

Existen otros puntos que se pueden explorar en futuras investigaciones. Se puede tratar de comparar la veracidad de los resultados contra algoritmos que no brindan un alineamiento óptimo para poder obtener un porcentaje de aceptación y contrastarlo con la eficiencia de ambos; con el fin de validar cuales algoritmos son de utilidad a gran escala. Ademas, se pueden tomar los algoritmos usados en esta tesis y ejecutarlos con secuencias grandes y reales en un ambiente paralelo o en cluster para validar rendimiento.

Bibliografía

Backofen, R. (17 de May de 2011). *Sequence Alignment Needleman-Wunsch.* Obtenido de Uni Freiburg Bioinformatics: http://www.bioinf.uni-freiburg.de/Lehre/Courses/2011_SS/V_Bioinformatik_1/needleman-wunsch.pdf

Hirschberg, D. (1975). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 341-343. Recuperado el 6 de July de 2013

Huson, D. (2003). *Algorithms in Bioinformatics I.* Tübingen: Universitat Tübingen.

Jackson, B. N., & Aluru, S. (2001). *Pairwise Sequence Alignments.* Ames: CRC Press.

Meneses, E. (s.f.). *Diseño de Algoritmos II.* Obtenido de Oocities: http://www.oocities.org/emenesesr/recursos/disenoAlgoritmosII.pdf

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

NHGRI. (13 de October de 2011). *Terminación del Proyecto Genoma Humano: Preguntas más frecuentes*. Obtenido de National Human Genome Research Institute: http://www.genome.gov/11510905

Pinzon, A. M. (14 de August de 2010). *Alineamiento: Analisis computacional de secuencias.* Obtenido de EMBnet Colombia: http://bioinf.ibun.unal.edu.co/cbib/estudiantes/1-07/alineamiento.pdf

Rodriguez Tello, E. A. (30 de May de 2013). *Alineamiento de pares de secuencias.* Obtenido de Cinvestav Tamaulipas: http://www.tamps.cinvestav.mx/~ertello/bioinfo/sesion05.pdf

Santamaria, R. (10 de April de 2013). *Alineamiento de pares de secuencias.* Obtenido de Vis usal: http://vis.usal.es/rodrigo/documentos/bioinfo/temas/3_Alineamientos%20de%20pares.pdf

Transparency Market Research. (28 de November de 2012). *Global Bioinformatics Market Expected Reach USA 9.1 Billion in 2018*. Obtenido de FierceBiotech: http://www.fiercebiotech.com/press-releases/global-bioinformatics-market-expected-reach-usd-91-billion-2018

Wunsch, C., & Needleman, S. (18 de May de 1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 443-453. Recuperado el 6 de July de 2013, de http://es.wikipedia.org/wiki/Algoritmo_Needleman-Wunsch

Xumari, G. (1967). *Introduction to dynamic programming.* New York: Wilwy & Sons Inc.