INSTITUTO TECNOLÓGICO DE COSTA RICA

ÁREA ACADÉMICA DE INGENIERÍA
EN COMPUTADORES

CURSO: CE-4302 ARQUITECTURA DE COMPUTADORES II

Paralelización de Algoritmos de Alineación de Secuencias por medio de Arquitecturas SIMD (extensiones SIMD y GPU)

Profesor:
Ing. Jeferson González Gómez, M.Sc

Estudiante:

Esteban A. Sanabria Villalobos 2015070913

09 de Junio, 2019

I. Introducción

En biología computacional, la alineación de secuencias es una preocupación prioritaria y muchos métodos se han desarrollado para resolver los problemas relacionados con la alineación de secuencias para aplicaciones biológicas. Needleman y Wunsch (Alunamiento Global) desarrollaron el conocido algoritmo de programación dinámica para la resolución al problema de alineación global de pares. Un algoritmo similar fue propuesto por Smith y Waterman (Alineamiento Local) para resolver el par local al problema de alineación. Además del problema de alineación por pares, el problema de alineación de secuencias múltiples también fue aclarado y se desarrollaron varios métodos para obtener la solución óptima. Este proyecto se centro en los métodos de comparación de dos secuencias, Global y Local, debido a su nivel relativamente sencillo y a su cualidad de ser un futuro punto de partida para la aplicación de los resultados obtenidos en la alineación de secuecnias multiples.

El proyecto consistió de la realización de un análisis de consumo de tiempo y memoria en cuanto al rendimiento de los algoritmos utilizados en el alineamiento de secuencias de gran extensión, las cuales en nuestro caso procedieron de la base de datos AmtDB (Ancient mitochondrial DNA database), la cual nos proveyó una amplia gama de secuencias de ADN proveniente de multiples organismos a través de la historia; las secuencias serán procesadas por los diferentes métodos de alineamiento, alineamiento Global y Local, sin y con optimizaciones. Así mismo, la realización e implementación del Benchmark, se encargara de ejecutar múltiples casos de prueba con secuencias de diversas longitudes y contra cada uno de los algoritmos de alineamiento mencionados en cada una de sus versiones, para de esa forma recompilar, promediar y brindar un resultado representativo de si es posible mejorar el desempeño y productividad de procesos bioinformáticos que utilicen como parte de su desarrollo este tipo de procesos de alineamiento para la obtención de información relevante en el área de la biología genética o molecular.

A todo esto, el problema que se desea resolver radica en la cantidad de tiempo y memoria que se requiere a la hora de procesar grandes volúmenes de datos presentes en las diversas áreas de estudio y aplicación de la Bioinformática.

II. Desarrollo de metodología

Primeramente, retomemos la metodología de desarrollo planteada en la propuesta inicial del presente proyecto y tomando esta como punto de partida seguiremos con el desarrollo de cada uno de los pasos a lo largo del camino de esta investigación y de los productos que cada una de las acciones planteadas dio como resultado a esta investigación.

II-A. Repaso de Metodología Planteada

Para la metodología a seguir para el desarrollo y obtención de datos de esta investigación nos basamos en un metodología experimental, la cual partiría desde el punto de selección de multiples cadenas de ADN de la base de datos de AmtDB, el procesamiento de estas cadenas para generar sub cadenas modificadas y de diferentes longitudes, se procedería a desarrollar los códigos de procesamiento de las secuencias en el lenguaje de programación de C en cada una de sus versiones (secuencial, extensión SIMD, CUDA); seguidamente, se procederá a desarrollar el Benchmark especializado para la tabulación y medición de los tiempos de ejecución y niveles de memoria utilizados en cada uno de los casos de prueba.

II-A1. Recolección y preparación de datos: Se utilizara la base de datos AmtDB para la sección manual de los archivos que contiene las secuencias. Se seleccionaran un conjunto de 15 archivos poseedores de secuencias de ADN de un largo aproximado de 16 000 caracteres cada uno, de forma que cada uno de los 15 archivos sean seleccionados de diferentes locaciones geográficas y que no provengan del mismo país. Cada uno de estos archivos poseerá la extensión *.fa, correspondiente a la extensión de FASTA. Estos archivos serán obtenidos durante tiempo de ejecución de la base de datos por medio de su dirección URL, por lo tanto, la única referencia a estas será un archivo de texto (*.txt) con las URL, una por linea en el archivo.

Estos archivos serán utilizados tanto en las versiones secuenciales de los dos algoritmos, como en las versiones SIMD. Para el caso del algoritmo de alineamiento Global (GSA) no se realizara ningún procedimiento de preparación de secuencias previo a su uso. Para el método local (LSA) se procederá por medio del método de ”shot gun”, reduciéndola a segmentos de menor longitud, para posteriormente, eliminar de

forma aleatoria algunos de estos segmentos y unir los restantes.

II-A2. Desarrollo de los algoritmos: Para esta investigación se trabajara sobre la implementación y optimización de dos algoritmos de alineamientos de secuencias (GSA y LSA) por medio de la utilización de herramientas, bibliotecas y hardware de tipo SIMD, utilizando la paralelización como método de optimización de procesamiento para el ahorro de tiempo, y mejor uso de la memoria por medio de buenas practicas de programación a la hora de ubicar datos en memoria, tomando en cuanta la forma en que el lenguaje de C ubica sus datos y estructuras en memoria.

Para cada uno de los algoritmos seguiremos la siguiente metodología:

1 - Investigar la teoría matemática detrás del algoritmo.

2 - Establecer el conjunto de ecuaciones matemáticas que definen los valores dentro de la matriz de puntajes a ser generada por cada algoritmo.

3 - Elaborar a partir de las ecuaciones previas los algoritmos necesarios para la implementación secuencial del algoritmo en cuestión.

4 - Realización de pruebas con secuencias pequeñas de ejemplo, para confirmar el funcionamiento del algoritmo.

5 - Basado en la teoría estudiada y las ecuaciones establecidas, deducir un nueva teoría de procesamiento de secuencias, sin modificar la logica detrás de cada método, para su paralelización, tomando en cuenta las herramientas SIMD. Considerando independencia de datos, matrices y procesos.

6 - Elaborar un conjunto de ecuaciones o pseudocódigo que brinden un modelo de programación para la implementación de las paralelizaciones.

7 - Elaborar a partir del modelo obtenido en el punto anterior la implementación del código de alineamiento en su segunda versión con extensiones SIMD (SSE).

8 - Realización de pruebas con secuencias pequeñas de ejemplo, para confirmar el funcionamiento del algoritmo en su nueva versión.

9 - Elaborar a partir del modelo obtenido en el punto 6, la implementación del código de alineamiento en su tercera versión para GPU's con CUDA.

10 - Realización de pruebas con secuencias pequeñas de ejemplo, para confirmar el funcionamiento del algoritmo en su nueva versión.

II-A3. Desarrollo del Benchmark: Se implementara un programa de evaluación el cual, tomara el archivo de texto con las URL's a cada una de las 15 secuencias, tomara cada una de las versiones de los algoritmos y los ejecutara dando como entrada de estos, combinaciones pre establecidas de las secuecnias de pruebas. Este tomara los tiempos de ejecución justo antes de iniciar el proceso de alineamiento y justo después de que este termine, para seguidamente realizar la resta entre el $tiempo_final - tiempo_inicial = tiempo_total$. Así mismo, se medirá el uso de memoria al iniciar la ejecucíon del algoritmo, al finalizar este y en momento considerados críticos en cuanto a la cantidad de datos que pondrían existir en determinado momento para cada algoritmo, esta medición intermedia de la memoria será realizada evaluando cada uno de los procesos y seleccionando un punto en común entre las diferentes versiones de los algoritmos de alineamientos, para así, mantener un estándar de medición y reducir las fuentes de error que se puedan producir por la mala toma de datos entre las diferentes pruebas.

II-A4. Automatización: Se procederá a realizar la creación de la biblioteca de utilidades básicas compartidas entre los algoritmos y sus versiones. Seguidamente, y de ser necesario la construcción de una biblioteca que brinde las utilidades de medición y ejecución del benchmark o bien de los algoritmos y sus versiones implementadas. Para finalmente, concluir esta sección con la elaboración de los archivos de compilación de cada uno de los recursos necesarios, de forma que el proceso de realización de pruebas y obtención de mediciones sea un proceso sencillo y fácil de replicar para cualquier otra persona que desea utilizar el código aquí desarrollado para la reproducción de los resultados.

II-A5. Presentación de Resultados: Los resultados a ser obtenidos por la ejecución de cada una de la versiones de los algoritmos serán almacenados y representados de una forma accesible y legible para los lectores. Tal que, estos se almacenaran al final de la ejecución en archivo de texto plano, con un formato por definir (posiblemente JSON).

II-B. Resultados de la Metodología

A continuación, se explica como se llevaron a cabo cada uno de los pasos de la metodología anteriormente mencionada y los productos obtenidos de cada uno de ellos.

II-B1. Recolección y preparación de datos: Como se planteo inicialmente, se utilizo la base de datos AmtDB para la sección manual de los archivos que contiene las secuencias. Se seleccionaran un conjunto de 17 archivos poseedores de secuencias de ADN de un largo aproximado de 16 000 caracteres cada uno. Cada uno proveniente de un país diferente como se observa en la Tabla 1 y Fig.1 . Cada uno de estos archivos posee la extensión *.fa. Estos archivos están siendo obtenidos durante tiempo de ejecución de la base de datos por medio de su dirección URL, almacenada en el archivo de texto usrls.txt, con las 17 URL por linea como se ve en la Fig.2 . Los archivos descargados durante la ejecución están siendo utilizados en las versiones secuenciales de los dos algoritmos, como en las versiones SIMD. Para el caso del algoritmo de alineamiento Global (GSA) se están utilizando las cadena según y como se obtiene de archivo *.fa correspondiente. Para el método local (LSA) se esta aplicando un proceso de división de la cadena (shot gun), reduciéndola a segmentos de menor longitud, para procesarla en los métodos LSA contra su original y obtener las similitudes entre estas.

II-B2. Desarrollo de los algoritmos: Para la implementación y optimización de los dos algoritmos de alineamientos de secuencias GSA y LSA, por medio de la utilización de herramientas, bibliotecas y hardware de tipo SIMD, se siguió la metodología presentada anteriormente, pero dado que para cada uno de los algoritmos a implementar se requería seguir el mismo conjunto de pasos y que el estudio de los casos de optimización se pueden reducir al estudio único del nivel de independencia entre los datos de la matriz

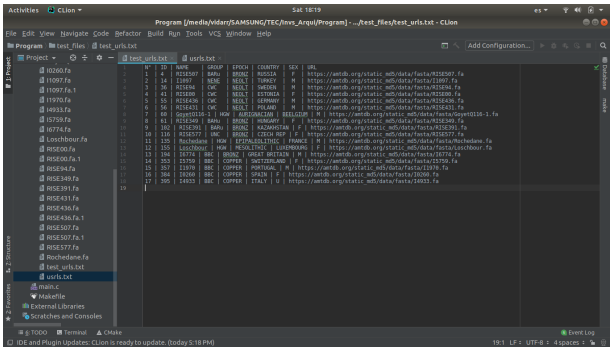

Figura 1. Archivo txt de descripción de las cadenas de ADN seleccionadas.

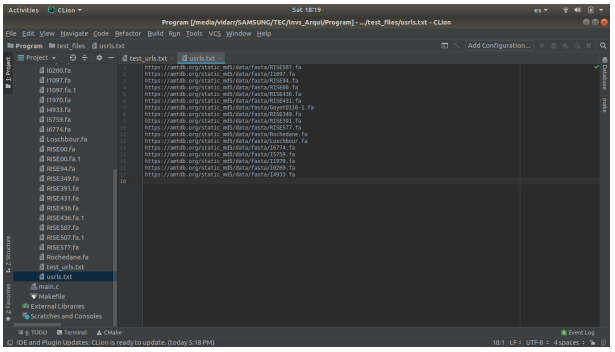

Figura 2. Archivo usrls.txt con las URL's de las cadenas de ADN

de puntajes utilizada y como los cálculos se pueden realizar de forma paralela sobre un conjunto de datos dependientes entre si, es que a continuación se presenta la metodología de esta sección como un estudio funcional, matemático y en seudo código.

Para el caso del estudio del algoritmo de Alineamiento Global se tiene la matriz de la Fig. 03 en la cual se lleva a cabo el alineamiento de las secuencias s='AAAC' y t='AGC', donde si 's' es tamaño 'm' y 't' es de tamaño 'n', se debe crear una matriz de tamaño (m+1) x (n+1). A demás, el primer renglón y la primera columna tienen múltiplos de penalidad por gap (—2) ya que solo hay alineamiento posible si una de las hileras está vacía. Es decir agregue tantos espacios como caracteres hay en la otra secuencia. El valor o score del alineamiento será -2*k, donde k es el largo de la secuencia no vacía.

- La clave se encuentra en el cálculo de las otras celdas (i, j)

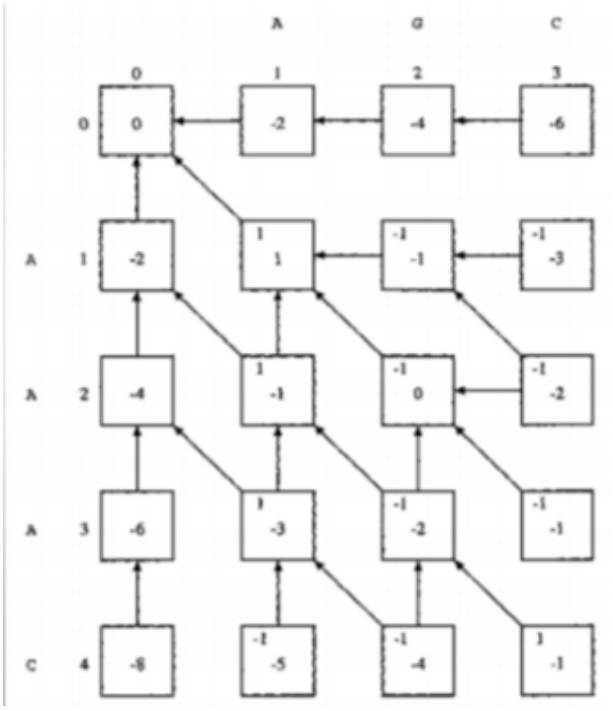

Figura 3. Ejemplo de matriz de alineamiento

- Se observan solo las posiciones previas: (i—1,j), (i—1,j—1) y (i, j—1)
- Solo hay tres formas de obtener el alineamiento entre s[1..i] y t[1..j]

$$a[i,j] = \max \begin{cases} a[i,j-1] - 2 \\ a[i-1,j-1] + p(i,j) \\ a[i-1,j] - 2. \end{cases}$$

Figura 4. Formula de selección de puntaje Global

- Costo del Algoritmo : $O(n^2)$
- Donde $P(i,j)$ corresponde a la función de comparación de entre los caracteres de las cadenas bajo comparación en la posición *i* y la posición *j*
- Seudo Código del Alineamiento Global

Seguidamente, para el estudio del caso de estudio del Alineamiento Local se toma como base lo ya visto para el alineamiento de tipo Global, pero con las siguientes variaciones:

Algorithm *Similarity*
input: sequences s and t
output: similarity between s and t
$m \leftarrow |s|$
$n \leftarrow |t|$
for $i \leftarrow 0$ **to** m **do**
  $a[i, 0] \leftarrow i \times g$
for $j \leftarrow 0$ **to** n **do**
  $a[0, j] \leftarrow j \times g$
for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow 1$ **to** n **do**
    $a[i, j] \leftarrow \max(a[i-1, j] + g,$
    $a[i-1, j-1] + p(i, j),$
    $a[i, j-1] + g)$
return $a[m, n]$

Figura 5. Seudo Código de Alineamiento Global

- Se requiere igual una matriz de (m+1) x (n+1). El primer renglón y la primera columna se llenan con Ceros.
- La interpretación varía: cada entra (i,j) contiene el score más alto entre un sufijo de s[1..i] y t[1..j]

$$a[i,j] = \max \begin{cases} a[i,j-1] + g \\ a[i-1,j-1] + p(i,j) \\ a[i-1,j] + g \\ 0, \end{cases}$$

Figura 6. Formula de selección de puntaje Alineamiento Local

- se busca el valor más alto resultante en toda la matriz, ese será el valor más alto óptimo del alineamiento local
- se siguen las flechas para determinar dicho alineamiento hasta que no hayan flechas o se llegue a Cero
- algunas veces pueden interesar también otros alineamientos locales a partir de algún valor de referencia

A continuación, se presenta en la Fig.8 la principal diferencia entre el alineamiento Global y el Local, donde la diferencia radica en que el algoritmo global considera todos los caracteres aun cuando estos no hacen match, pero el alineamiento global solo toma en consideración las secciones de las secuecnias que tiene el mayor número de alineamiento.

Input: two sequences x and y with length n and m, respectively; scoring matrix s, gap penalty d
Output: optimal local alignment and its score

BEGIN INITIALIZATION
$S(i,0) = S(0,j) = 0$ for $0 \le j \le m$ and $0 \le i \le n$
END INITIALIZATION

BEGIN PROCEDURE
for $1 \le i \le n$ **do**
for $1 \le j \le m$ **do**
$a(i-1,j-1) = S(i-1,j-1) + s(x_i, y_j)$, $a(i-1,j) = S(i-1,j) - d$,
$a(i,j-1) = S(i,j-1) - d$
$S(i,j) = \max\{0, a(i-1,j-1), a(i-1,j), a(i,j-1)\}$
if $S(i,j) > 0$ **then** $B(i,j) = \arg\max\{0, a(i-1,j-1), a(i-1,j), a(i,j-1)\}$ **else**
$B(i,j) = (-1,-1)$
end for
end for
$(i,j) = \arg\max\{S(i,j) \mid 1 \le i \le n,\ 1 \le j \le m\}$
print "Score: " $S(i,j)$

while $S(i,j) \neq 0$ **do**

$$\textbf{if } B(i,j) = \begin{cases} (i-1,j-1) & \textbf{then print } \begin{smallmatrix} x_i \\ y_j \end{smallmatrix} \\ (i-1,j) & \textbf{then print } \begin{smallmatrix} x_i \\ - \end{smallmatrix} \\ (i,j-1) & \textbf{then print } \begin{smallmatrix} - \\ y_j \end{smallmatrix} \end{cases}$$

$(i,j) = B(i,j)$
end while
END PROCEDURE

Figura 7. Seudo Código del Alineamiento Local

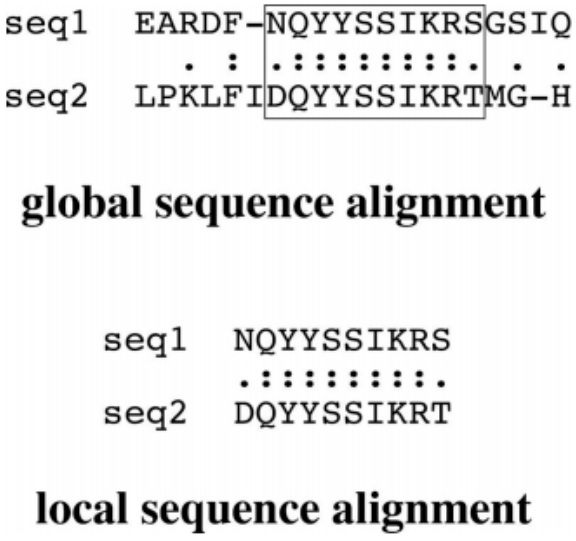

Figura 8. Diferencia entre Alineamiento Global y Local

Ahora, para el estudio e implementación de las optimizaciones realizando uso de la paralelización a nivel de nivel de datos en los algoritmos de alineamiento de debe tener en cuenta que:

El problema del alineamiento se puede resolver utilizando un espacio de memorización bidimensional F (matrices). Donde:

- El elemento directamente arriba de F[i,j] es F[i-1, j]
- El elemento directamente a la izquierda de F[i,j] es F[i, j-1]
- El elemento directamente a la noreste de F[i,j] es F[i-1, j-1]

Esta dependencia crea una computación diagonal, es decir una cadena a través del espacio del problema, formando un frente de onda. La dependencia y la dirección de cálculo se muestran en la siguiente figura.

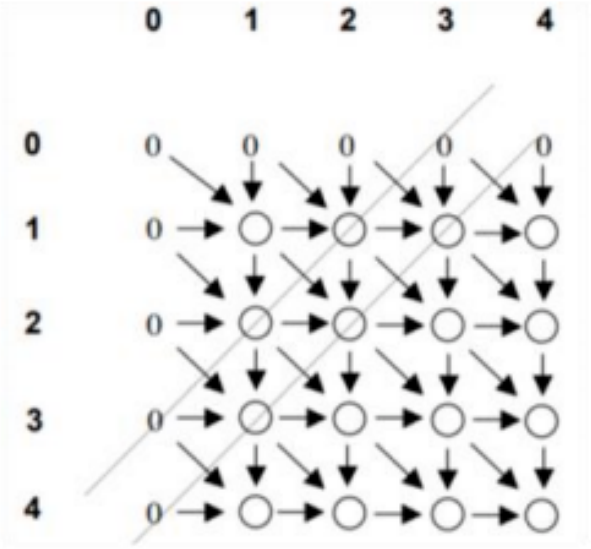

Figura 9. Diferencia entre Alineamiento Global y Local

Cuando se utiliza el algoritmo de programación dinámica para alinear dos secuencias, el proceso puede verse como los cálculos de puntuación en una matriz bidimensional. Al utilizar el algoritmo de Hirschberg, esta matriz bidimensional se dividirá en una matriz directa y una matriz inversa por igual. En la matriz de avance, la ruta de avance es calcular las puntuaciones desde la parte superior izquierda a la derecha y hacia abajo. De manera similar, en la matriz inversa, la ruta inversa es calcular las puntuaciones de derecha a abajo a la parte superior izquierda. Los cálculos de la matriz directa y la matriz inversa se pueden realizar de forma simultánea.

De forma tal que los cálculos de la matriz pueden ser procesados por bloques principalmente en el caso de el uso de las GPU, ya que como se muestra en la Fig.9, los datos son sumamente dependientes, tal que en el enfoque de programación con extensiones SIMD se programara una versión que haga uso de las capacidades de manipulación de vectores y el factor de ”seudo-encadenamiento”de los datos para que en forma teórica, se disminuya la cantidad de tiempo necesaria para el procesamiento de las secuencias de caracteres.

Para el caso de las Optimizaciones por GPU se debe tomar una orientación un poco más variada, debido a esta misma dependencia de los datos, la programación para GPU no puede tener este enfoque de encadenamiento porque no se estaría utilizando de forma correcta la arquitectura y en el más probable de los casos la ejecucíon se vuelva más lenta debido a la espera entre bloques y threads a la espera de los resultados previos de la matriz. Es por esto que se sigue una orientación por bloques y por hilos de ejecución, la cual se presenta en la Fig.10, donde se ve que se procesa la matriz por bloques y dependiendo del tamaño del bloque se crean diferentes threads que se encargan de procesar los datos en una forma más independiente y paralela que con el enfoque del encadenamiento.

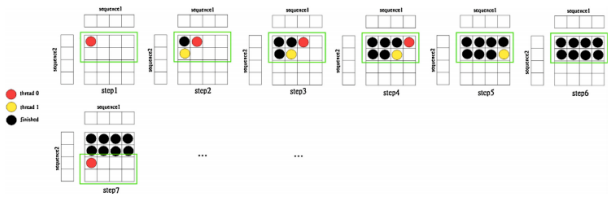

Figura 10. Procesamiento de Alineamientos por bloques y threads

Para la visualización de la implementación de estas optimizaciones el código de cada una de las versiones puede visitar el repositorio del proyecto en **Github**: *https://github.com/Esteasv31/Alignment_Optimization_SIMD.git*

II-B3. Desarrollo del Benchmark: Se implemento un programa de evaluación el cual, toma el archivo de texto con las URL's a cada una de las 17 secuencias, por medio de la ejecución de comandos de Linux en C se ejecuta el comando *wget* con las banderas *-P ../../test_files* para indicarle la ubicación donde se descargaran los archivos de prueba con las secuecnias de ADN. Posteriormente, este tiene una rutina preprogramada para la ejecución de cada uno de los casos de prueba iniciando por los casos secuenciales del GSA y LSA, posteriormente ejecuta las versiones con extensiones SIMD de cada uno de estos algoritmos y para finalizar lleva a cabo la ejecución de las versiones de CUDA para GPU de los mismos algoritmos, recolectando los datos de tiempo de ejecución de cada una de estas ejecuciones de forma que toma los tiempos de ejecución justo antes de iniciar el proceso de alineamiento y justo después de que este termine, para realizar la resta entre el $tiempo_final - tiempo_inicial = tiempo_total$. Así mismo, mide el uso de memoria al iniciar la ejecucíon del algoritmo, al completar la matriz de puntajes, al realizar el proceso de reconstrucción del alineamiento y al finalizar este; de forma que se puede comparar el uso de la memoria de los algoritmos en cuatro secciones diferentes de la ejecución de estos. Ya que las llamadas al sistema para medir la cantidad de memoria producen un efecto de error en la medición de los tiempos de ejecución se crearon dos versiones completamente idénticas de los métodos con la pequeña diferencia de que uno realiza su ejecución de forma continua y lo más limpia posible de procesos intermedios que afecten la medición de tiempo, y la segunda igual de limpia pero orientada a la medición de memoria. Esto con el fin de eliminar posibles factores de error entre las mediciones a realizar y disminuir los margenes de error a la hora de comparar los resultados finales. Finalmente, este software de evaluación coloca los resultados de las ejecuciones en un archivo de resultados identificado con la fecha y hora de la ejecucíon; y el contenido de este en el formato que se plantea en la sección de *Presentación de Resultados* de este documento.

II-B4. Automatización: Para esta parte del proceso de desarrollo y de la metodología se procedió a la realización de un conjunto de Makefiles que están estructurados para realizar el proceso de compilación completo del proyecto y de la unión de cada una de sus parte. Como se menciono inicialmente, se pretendía crear una serie de bibliotecas a partir de cada una de las versiones y códigos generados para esta investigación de forma que a futuro las replicaciones de los resultados acá obtenidos fueran de sencilla repetición para cualquier persona, de forma que como parte del proceso de automatización de crearon los scripts necesarios para la construcción de las bibliotecas estáticas

de cada una de las versiones, así como de las funciones de utilidades creadas y del benchmark en sí, para la reproducción total de la investigación y corrida automatizada del proceso de pruebas.

El Makefile en el top del proyecto es el encargado de compilar la totalidad del proyecto y generar todos y cada uno de los ejecutables y archivos de bibliotecas creados y mencionados, dentro de la estructura misma del proyecto, todo esto por medio del comando en terminal *make*. Una vez finalizado este proceso se puede proceder a utilizar los compilados o bien a ejecutar las pruebas automatizadas por medio del comando *make run_test*. Finalmente, el proceso de automatización termina con la implementación del comando *make clean* encargado de limpiar la estructura del proyecto de todos los archivos ejecutables generados, archivos de prueba descargados, entre otros.

II-B5. Presentación de Resultados: Finalmente, como parte de la metodología y como resultado final de esta se logro generar un archivo de resultados que representa cada una de las pruebas llevadas a cabo con el Benchmark, donde se indican los métodos, las entradas de estos, el tiempo de ejecución y las cuatro mediciones de memoria en los diferentes punto seleccionados. El formato de este archivo iniciando por el nombre, corresponde a la fecha y hora de ejecución de la prueba de forma que cada una de las corridas tiene un único archivo de ejecución, este archivo es de tipo texto por lo cual la extensión de este es de tipo *.txt, y cada uno de estos archivos se almacena en la carpeta de *results_files*. Para finalizar, el formato interno de este documento es un string en formato JSON donde las llaves tiene el formato *exe_*, donde corresponde al número de ejecución y donde el valor asociado a cada una de estas llaves es otro string en formato JSON con las llaves *method* (método utilizado), *entry_1* (input 1), *entry_2* (input 2), *exe_time* (tiempo total de ejecución), *init_mem* (estado de la memoria al inicio de la ejecución), *full_mem*(estado de la memoria al completar la matriz de puntajes), *build_mem*(estado de la memoria al completar la reconstrucción del alineamiento), *end_mem* (estado de la memoria al final de la ejecución). Ejemplo:

$\{'exe\_1' :' \{'method' :' ','entry\_1' :' ',$
'entry_2' : ' ', 'exe_time' : ' ', 'init_mem' : ' ', 'full_mem' : ' ', 'build_mem' : ' ', 'end_mem' : ' ' $\}'$,$'exe\_2' :' \{'method' :' ','entry\_1' :' ',$
$'entry\_2' :' ',$
'exe_time' : ' ', 'init_mem' : ' ', 'full_mem' : ' ', 'build_mem' : ' ', 'end_mem' : ' ' $\}', ..., 'exe\_n' :'$
$\{'method' :' ','entry\_1' :' ','entry\_2' :' ',$
'exe_time' : ' ', 'init_mem' : ' ', 'full_mem' : ' ', 'build_mem' : ' ', 'end_mem' : ' ' $\}'\}$

III. Conclusión

En cuanto a lo que respecta a la metodología planteada y la desarrollada se puede concluir que la orientación experimental calza de gran forma con la intención del proyecto, ya que a través del estudio del tema bajo investigación, la comprensión del mismo, la generación de nuevas teorías y la ejecución repetida de eventos para establecer un conjunto de resultados estandarizados y que brinden una representación certera de los fenómenos en cuestión fueron procesos esenciales en el desarrollo del presente proyecto. Seguidamente, la metodología planteada en la mayoría de los puntos logró su objetivo aunque por cuestiones de tiempo y organización entre tareas muchas de ellas no tiene el nivel de detalle deseado al inicio del proyecto. Finalmente, la metodología sirvió de gran forma para comprender la teoría detrás de cada una de las pruebas y como base esencial a la hora de analizar y generar conclusiones acerca de los resultados obtenidos y porque estos se comportan a como lo hacen.