Universitat Autònoma de Barcelona

Departament d'Arquitectura de Computadors i Sistemes Operatius

Màster en Computació d'Altes Prestacions

Alineamiento Múltiple de Secuencias con T-Coffee: Una Aproximación Paralela

Memoria del trabajo de investigación del "Máster en Computación de Altas Prestaciones", realizada por **Yandi Naranjo Basalo**, bajo la dirección de **Porfidio Hernández Budé** Presentada en la Escuela Técnica Superior de Ingeniería (Departamento de Arquitectura de Computadores y Sistemas Operativos)

Barcelona, Julio 2009

41746 - Iniciació a la recerca i treball fi de màster

Máster en Computación de Altas Prestaciones

Curso **2008-09**

Título: Alineamiento Múltiple de Secuencias con T-Coffee: Una Aproximación Paralela

Autor: Yandi Naranjo Basalo

Director: Porfidio Hernández Budé

Departamento Arquitectura de Computadores y Sistemas Operativos

Escuela Técnica Superior de Ingeniería (ETSE)

Universidad Autónoma de Barcelona

Firmado

Autor Director

Agradecimientos

Quiero darles un especial agradecimiento a Porfidio Hernández Budé y Antonio Espinosa por todo el esfuerzo y la dedicación que han tenido para poder desarrollar esta investigación. Un agradecimiento a los que me han dado la oportunidad a través de un proyecto del departamento de Arquitectura de Computadores y Sistemas Operativos, ETSE, UAB, de formarme como personal investigador. Agradecerle además a los profesores del departamento, a todos…

A los compañeros del departamento, en especial a Genaro, Claudia, Ronald, José Ramón, Rodrigo, Gonzalo, Moni, Álvaro….todos

Un beso y agradecimiento por la ayuda incondicional, el apoyo y el amor que me ha brindado mi esposa Maria, y mi hijita Laura que a pesar de no ayudarme de forma directa es mi motor impulsor….mis padres que desde lo lejos me apoyan como siempre y mi hermano que nunca falta….

Y de forma general un agradecimiento a todos aquellos que me han brindado una mano y han hecho posible que hoy esté donde estoy……

A mi hijita Laura,
mi esposa Maria,
mis Padres y hermano…

Resumen

Las aplicaciones de alineamiento múltiple de secuencias son prototipos de aplicaciones que requieren elevada potencia de cómputo y memoria. Se destacan por la relevancia científica que tienen los resultados que brindan a investigaciones científicas en el campo de la biomedicina, genética y farmacología. Las aplicaciones de alineamiento múltiple tienen la limitante de que no son capaces de procesar miles de secuencias, por lo que se hace necesario crear un modelo para resolver la problemática. Analizando el volumen de datos que se manipulan en el área de las ciencias biológica y la complejidad de los algoritmos de alineamiento de secuencias, la única vía de solución del problema es a través de la utilización de entornos de cómputo paralelos y la computación de altas prestaciones. La investigación realizada por nosotros tiene como objetivo la creación de un modelo paralelo que le permita a los algoritmos de alineamiento múltiple aumentar el número de secuencias a procesar, tratando de mantener la calidad en los resultados para garantizar la precisión científica. El modelo que proponemos emplea como base la clusterización de las secuencias de entrada utilizando criterios biológicos que permiten mantener la calidad de los resultados. Además, el modelo se enfoca en la disminución del tiempo de cómputo y consumo de memoria. Para presentar y validar el modelo utilizamos T-Coffee, como plataforma de desarrollo e investigación. El modelo propuesto pudiera ser aplicado a cualquier otro algoritmo de alineamiento múltiple se secuencias.

Abstract

The multiple sequence alignment applications are of those requiring high computing potency and memory. This kind of applications has a hugh scientific importance due to their contribution to investigations in biomedicine, genetics and pharmacology for example. However, these applications are limited in the number of sequences they can analyze at once, being incapable to process thousand of sequences and justifying the need for a model to solve this problem. If we analyze the amount of data that is manipulated in biological investigations, and the complexity of the algorithms for sequence alignment, we realize that the only way to solve the problem is through the use of parallel and high performance computing. The research presented in this work, is about a parallel model allowing the increase of input sequences to the multiple alignment algorithms trying to maintain the quality in the final results to guarantee the scientific precision. The model proposed uses the clustering of input sequences as a base, employing biological information to maintain the quality of the results, and also diminishing the computing time and memory consumption. We used the T-Coffee algorithm as a development and research platform to present and validate the proposed model. We intend that our proposed model could be applied to any other algorithm for multiple sequence alignment.

Tabla de Contenidos

Índice de Figuras

CAPITULO I

Introducción

En este capítulo se presentará una visión general del problema que presentan las aplicaciones de alineamiento múltiple de secuencias, y la importancia que tiene resolverlo para las investigaciones científicas de la salud. Primero se introducirán conceptos biológicos necesarios para comprender el problema. Se explicará en qué consiste y lo que significa el alineamiento de secuencias biológicas y qué métodos se emplean para intentar resolver el problema. Se mostrará que la única vía para solucionar el problema es la aplicación de la Computación de Altas Prestaciones. Finalmente, se presentarán los objetivos de la investigación y las aportaciones que supone.

I.1 Antecedentes

El estudio del cómputo paralelo en áreas relacionadas con las ciencias de la vida, la predicción meteorológica, geología y sísmica, entre otras disciplinas, ha sido una constante y ha crecido de manera significativa en la última década. Muchos de los retos a los que nos enfrentamos desde el punto de vista científico (Figura I.1), no pueden ser abordados sin la potencia de cómputo que nos proporcionan los supercomputadores. Actualmente, el cómputo paralelo es una de las alternativas más prometedoras a efectos de lograr mejorar la ejecución de las aplicaciones. La búsqueda de entornos de programación paralela que hagan transparente al usuario la complejidad que supone trabajar en nuevas arquitecturas, diseñar políticas de planificación y balanceo de tareas a nodos de procesamiento, diseñar herramientas que permitan afrontar problemáticas de tolerancia a fallos, entre otros problemas a resolver, configuran un nuevo escenario con multitud de problemas abiertos a los que dar solución.

Muchos de los actuales retos científicos han requerido de nuevos y novedosos recursos de cómputo principalmente por dos razones: ahorro de tiempo, y aumentar el tamaño de los problemas a tratar. Una de las formas de enfrentar problemáticas de esta índole es a partir del desarrollo de soluciones computacionales que permitan: tomar ventaja de recursos locales y disponibles en la red; ahorrar en costo a partir del uso de múltiples recursos computacionales

más económicos, en vez de costosos supercomputadores; y eliminar la limitación de la memoria finita de un computador, incrementada por el uso de múltiples computadores.

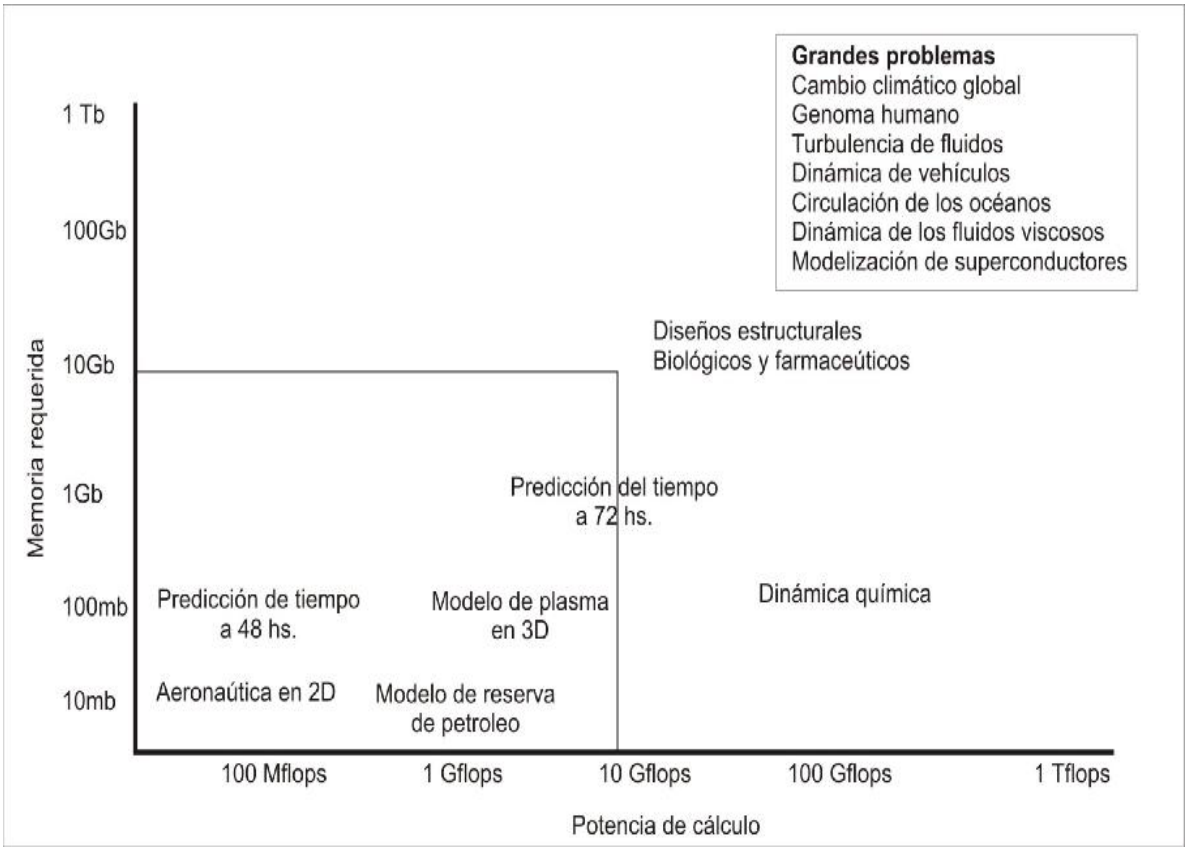


Figura I.1 Requerimientos computacionales de problemas complejos.

Entre las problemáticas representadas en la Figura I.1 destacan las aplicaciones con carácter biológico como prototipo de las que más potencia de cómputo y memoria requieren. El estudio de estrategias y modelos de análisis, se hace necesario para el desarrollo de aplicaciones paralelas en esta área.

Las aplicaciones computacionales con carácter biológico, están ganando una importancia capital para la obtención de conocimiento a partir del análisis de la información biológica que se obtiene hoy día, por la necesaria aplicación en el diseño de fármacos, en la obtención de vacunas, en el estudio de enfermedades hereditarias y de nuevas enfermedades. El análisis de la información biológica siempre va acompañado de experimentación en laboratorios, pero en los últimos años se está apoyando más en el estudios e investigación *in silico* (*hecho por ordenadores o a través de simulaciones computacionales*), para abaratar el costo en investigación. Los datos a analizar

en investigaciones biológicas *in silico* en casi toda su totalidad son: la secuencia de los genomas (*ADN*) y de las proteínas (Figura I.2). El objetivo de analizar este tipo de bio-moléculas, es comprender su funcionalidad biológica en los seres vivos.

ADN: Material genético de casi todos los organismos. Secuencia de nucleótidos (*molécula orgánica, subunidad del ADN y ARN*), cada uno con una de las posibles bases nitrogenadas (*compuestos orgánicos que forman los nucleótidos*): adenina (*representada por* A), timina (*representada por* T), citosina (*representada por* C) o guanina (*representada por* G). El alfabeto de estas secuencias consta de 4 letras, que se corresponden con las 4 posibles bases nitrogenadas (A, T, C, G).

ARN: Macromolécula compuesta por una secuencia de nucleótidos cada uno con una de las posibles bases: uracilo (*representada por* U), adenina (*representada por* A), guanina (*representada por* G) o citosina (*representada por* C) y que se ubican en la secuencia de forma complementaria a las del ADN que le sirve de molde. El alfabeto de estas secuencias consta de 4 letras, que se corresponden con las 4 posibles bases nitrogenadas (A, U, C, G). Dirige la síntesis de proteínas.

Proteína: Macromoléculas compuestas por una secuencia de aminoácidos (*molécula orgánica, subunidad de la proteína*), codificados en el DNA y traducidos según el código genético. El alfabeto de estas secuencias consta de 20 posibles letras, representando a cada uno de los 20 posibles aminoácidos (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y).

Secuencia: Cadena de caracteres representativos de una molécula, ya sea ADN o ARN cuyos alfabetos constan de 4 letras, o proteínas con un alfabeto de 20 letras. Cadena lineal, finita y ordenada de símbolos pertenecientes a un alfabeto.

Figura I.2 Definición de secuencias biológicas.

Para lograr entender la importancia tanto de las secuencias de *proteínas* como las secuencias de *ADN* y *ARN d*entro de la biología computacional, en primer lugar debemos identificar en qué nivel de análisis se sitúan. Para entenderlo recorramos mediante un ejemplo la organización jerárquica de las estructuras biológicas. En la Figura I.2 se han definido los términos de *proteínas*, *ADN* y *ARN*, por lo que comenzaremos por la estructura superior, hasta llegar al nivel molecular. Si observamos la Figura I.3, nos damos cuenta que los seres vivos están construidos de una manera jerárquica. Los organismos multicelulares como el *humano,* son sistemas de

órganos que están constituidos por tejidos formados por células compuestas por orgánulos subcelulares. Y en este punto entramos en el reino de la bioquímica, en el que los orgánulos subcelulares están formados por macromoléculas. Los seres vivos en general contienen, solamente unos pocos tipos diferentes de macromoléculas, entre las que se encuentran las *proteínas*, *ADN* y *ARN*.

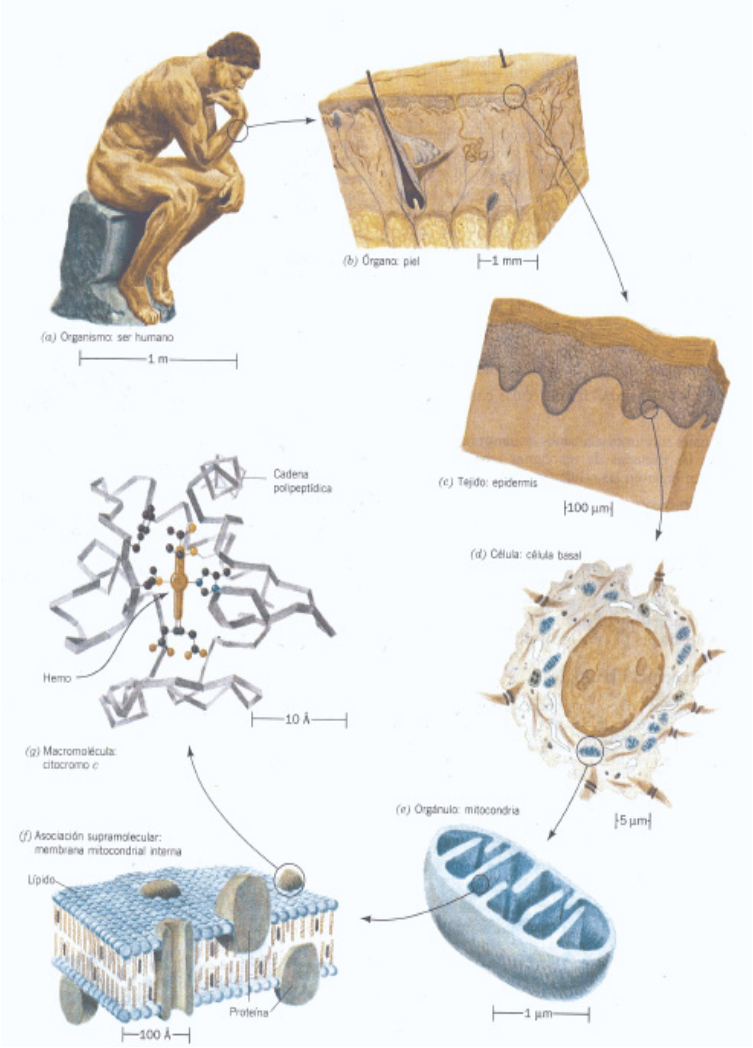


Figura I.3 Representación jerárquica de las estructuras biológicas.

Se conoce que la causa de la evolución de los organismos a través del tiempo, son los *cambios* en la secuencia de *ADN* de los mismos (*mutaciones*). De esta manera con un cambio a nivel molecular, se afectarán consecutivamente, los orgánulos, células, tejidos y órganos a los que pertenece la secuencia de A*DN* cambiada, y finalmente se observará un cambio a nivel de organismo. Todo bajo la premisa de organización jerárquica de los organismos antes expuesta.

Un ejemplo de la vida real es el caso de la enfermedad de anemia falciforme humana causada por afectaciones en la hemoglobina (Figura I.4). La Hemoglobina normal contiene ácido glutámico (*aminoácido representado por una* E *en la secuencia*) en una posición determinada de su secuencia; la hemoglobina falciforme tiene Valina (*aminoácido representado por una* V *en la secuencia*) en esa misma posición. Este cambio en la secuencia de la proteína se origina por cambios en la secuencia de ADN de la propia célula. Este simple cambio de un aminoácido por otro es suficiente para que la hemoglobina no funcione correctamente y se produzca la enfermedad.

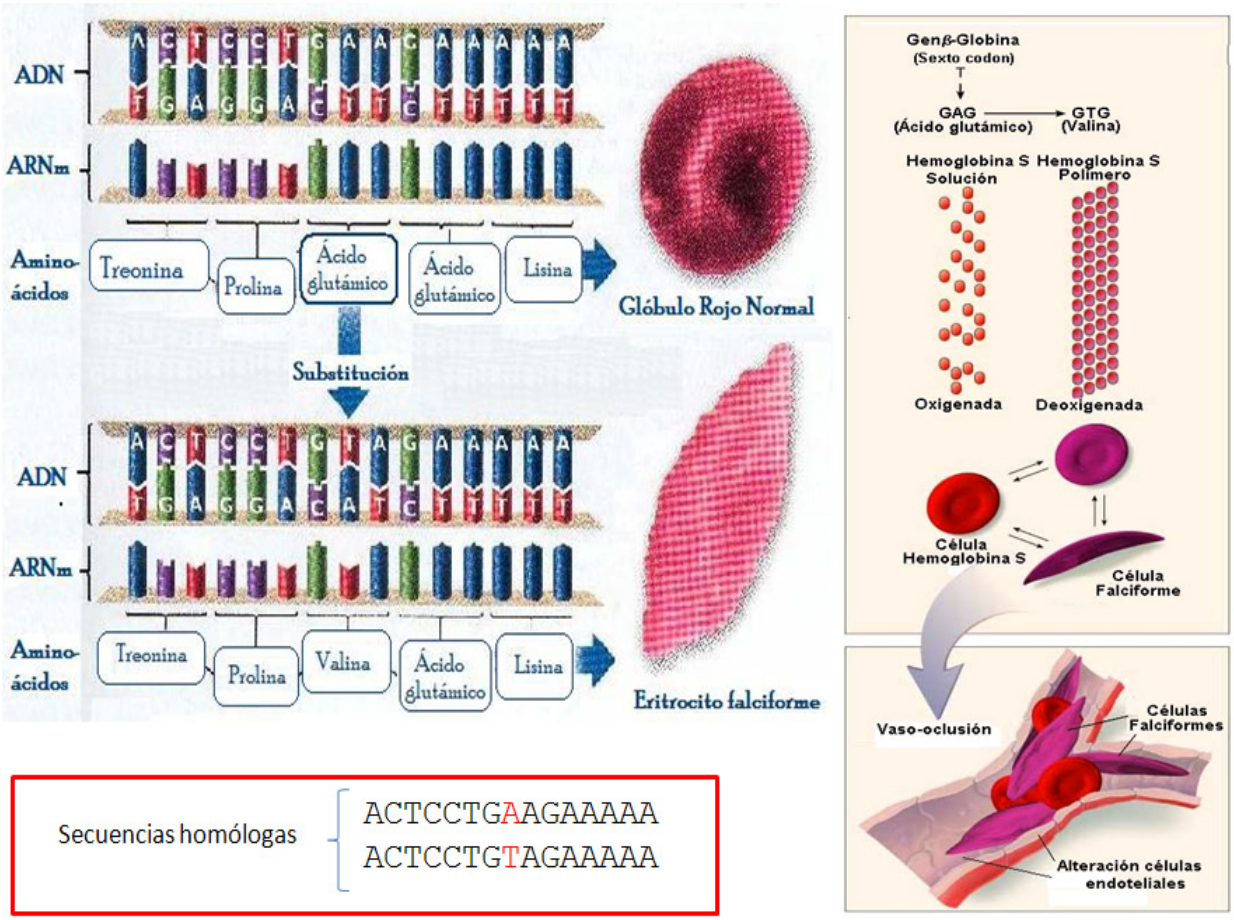


Figura I.4 Cambio de un aminoácido por error en la secuencia de la hemoglobina que origina la enfermedad de anemia falciforme.

En la parte inferior izquierda de la Figura 4, se muestran las dos secuencias de ADN de la hemoglobina (*la secuencia que expresa la enfermedad y la sana*), las dos secuencias solo se diferencian por una base nitrogenada (*el cambio de A por T*), y el cambio de esta base

nitrogenada provoca un cambio de aminoácido (*ácido glutámico por valina*) en la hemoglobina. Solo el cambio de una base nitrogenada en el código produce la enfermedad. Este ejemplo es la comparación de dos secuencias que tienen la misma función y presentan pocos cambios en las secuencias (*en este caso en una base nitrogenada*), clasificándose por este motivo en secuencias homólogas. Visto así, la secuencia genética es una buena base para empezar a analizar las funciones de los genes mediante las secuencias de las bases de datos, sin tener que hacerlo en laboratorios.

El ejemplo de la Figura I.4 evidencia una enfermedad presente a nivel molecular por un error en un aminoácido, visto desde dos secuencias homólogas, y que puede ser detectado por el análisis de la secuencia de proteína. El hecho de obtener conocimiento a partir de secuencias de proteínas y ADN, se extiende a lo largo de muchos casos, y además de permitir estudiar enfermedades desde este nivel tan básico de la vida, permite obtener nuevos conocimientos de la comparación a nivel molecular.

De acuerdo con la teoría de la evolución, las secuencias de ADN y las proteínas de los organismos contemporáneos se originaron por mutaciones, que han ocurrido en las secuencias de organismos ancestrales. La Figura I.5 muestra la evolución de fragmentos de secuencias genéticas, en formal de árbol con un ancestro común (*la secuencia de la parte superior del árbol*), del que van derivando las secuencias genéticas inferiores en el árbol, obtenidas por cambios en alguna de las bases. Por ese motivo, es posible encontrar que secuencias homólogas, tengan cierto grado de similitud. Siendo posible asignar un significado biológico a secuencias con función biológica desconocida, partiendo de secuencias con función biológica conocida que se parecen a las primeras (*realizando un análisis parecido al descrito en el ejemplo de la hemoglobina*). Es decir, se pueden estudiar funciones de los genes analizando bases de datos de secuencias mediante programas de búsqueda y comparación de secuencias.

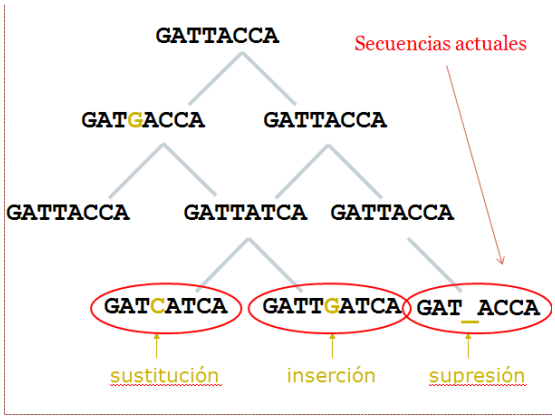


Figura I.5 Representación del proceso evolutivo de secuencias.

La forma de extraer información a partir de las secuencias biológicas es mediante la acción de comparar (*de buscar secuencias similares: por ejemplo una secuencia desconocida que pase a ser "familia" de otra cuya función está anotada, definiendo familia a secuencias cuya función biológica es la misma*). La comparación se puede realizar mediante métodos de alineamiento de secuencias, que lo que intentan es modelar procesos evolutivos, en el que varias secuencias se derivan de un ancestro común a través de cambios incrementales como se muestra en la Figura I.5, debido a errores en la replicación del ADN, mutaciones o daños. Los mecanismos que dan lugar a cambios en las secuencias pueden ser calificados como: sustitución (**ACGA** → **AGGA**), inserción (**ACGA** → **ACGGA**) y supresión (**ACGGA** → **ACGA**). Las dos últimas mutaciones provocan huecos o *gaps*.

I.2 Alineamiento de secuencias biológicas

El objetivo de los algoritmos de alineamiento de secuencias, es acomodar dos a más secuencias de tal manera que se alcance el máximo de coincidencias entre los elementos de las mismas. Los alineamientos se pueden clasificar en: globales (*cuando se pretende alinear las secuencias enteras, empleando tantos caracteres o símbolos de los extremos de las secuencias como sea posible*), y locales (*cuando se buscan porciones de las secuencias que presentan mayor cantidad de concordancias*). Los alineamientos de secuencia se resuelven a través de métodos computacionales de programación dinámica (*solución óptima, alineamiento global; Needleman & Wunsh, 1970; alineamiento local; Smith & Waterman, 1980*), o por algoritmos heurísticos (*solución sub-óptima*). El proceso de alineamiento de secuencia, es el responsable en gran medida de la demanda de cómputo de la mayoría de las aplicaciones biológicas.

El alineamiento de dos secuencias, por programación dinámica (*PD*, Figura I.6), se basa en la construcción de una matriz de valores, en la que se dispone, una de las secuencias en las filas y otra en las columnas. El tamaño de la matriz es de (M + 1)*(N + 1), donde M y N son las longitud de las secuencias a alinear. La primera fila y columna se corresponde con la asignación de un hueco (*gap*), que es representado en una de las dos secuencias, por la separación mediante una línea o guión (Figura I.6, *posición derecha superior*). A cada celda de la matriz se asigna un valor calculado de acuerdo a un esquema de puntuación, en el que se puntúan: la coincidencia, la no coincidencia y la introducción de un hueco (*gap*) en cualquiera de las secuencias (Figura I.6). Cada celda se calcula por el valor máximo entre: la celda diagonal superior izquierda más un valor de puntuación, la celda superior menos una penalización por introducir huecos (*gap*), y la celda izquierda menos una penalización por introducir huecos (*gap*). El valor de puntuación tanto para la coincidencia, como la no coincidencia y la penalización, define el esquema de evaluación del alineamiento. Una vez calculada la matriz de valores, se busca la ruta que mayor (*para el caso que la coincidencia se evalúe positiva*) o menor (*en el caso que la coincidencia se evalúe negativa*) valor obtenga en la matriz y esto se traduce en el alineamiento óptimo (Figura I.6). La complejidad del algoritmo por la forma en que se calcula es $O(L^2)$, siendo L el valor medio de la longitud de las secuencias.

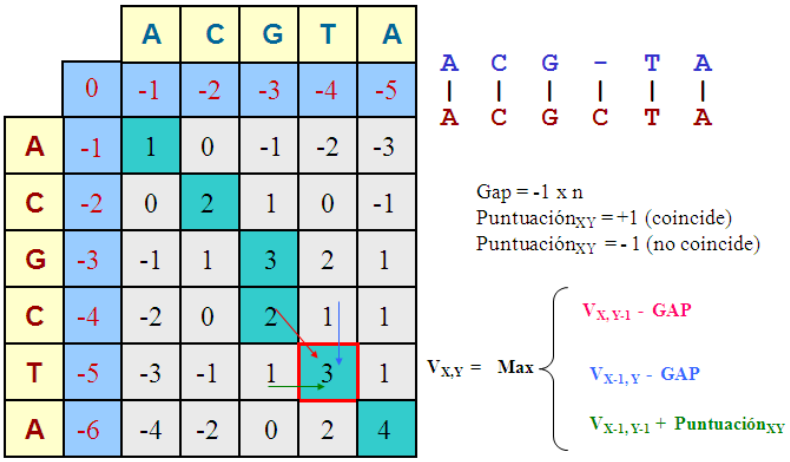


Figura I.6 Alineamiento de dos secuencias por PD.

Los algoritmos de alineamiento de secuencias, emplean ciertas métricas para determinar la calidad del alineamiento, una es el porcentaje de *similitud de secuencia*, que no es más que la fracción de elementos iguales alineados, entre la cantidad de elementos de la secuencia más corta. Otra de las métricas empleadas para representar la calidad del alineamiento es el *score*, determinada por una función de puntuación del algoritmo, para definir la bondad del alineamiento obtenido (*generalmente se devuelve un valor entre 0 y 100%, que mientras más elevado sea, mejor es el alineamiento*).

A menudo los esquemas de puntuación en el alineamiento de proteínas, utilizan ciertas matrices (*las matrices PAM y BLOSUM son las más comunes*) obtenidas a partir del análisis del proceso evolutivo de secuencias conocidas. Estas matrices reflejan la probabilidad de que los residuos (*aminoácidos*) hayan quedado alineados dado que provienen de un ancestro común, con respecto a la probabilidad de que se hayan alineado al azar. Las matrices *PAM* y *BLOSUM* tienen en cuenta un valor de puntuación por la sustitución de aminoácidos, proporcionando de esta forma más consistencia a los alineamientos de secuencias.

Dentro de la biología computacional, uno de los algoritmos que más consumen recursos (*memoria y tiempo de cómputo*) son los de alineamiento múltiple de secuencias (Chakrabarti *et al.,* 2006; Rezaei *et al.,* 2006). Tratar de obtener una solución óptima del alineamiento múltiple de varias secuencias explorando todo el espectro de posibilidades (*empleando técnicas de programación dinámica*), es una tarea hasta el momento imposible para un número no muy elevado de secuencias.

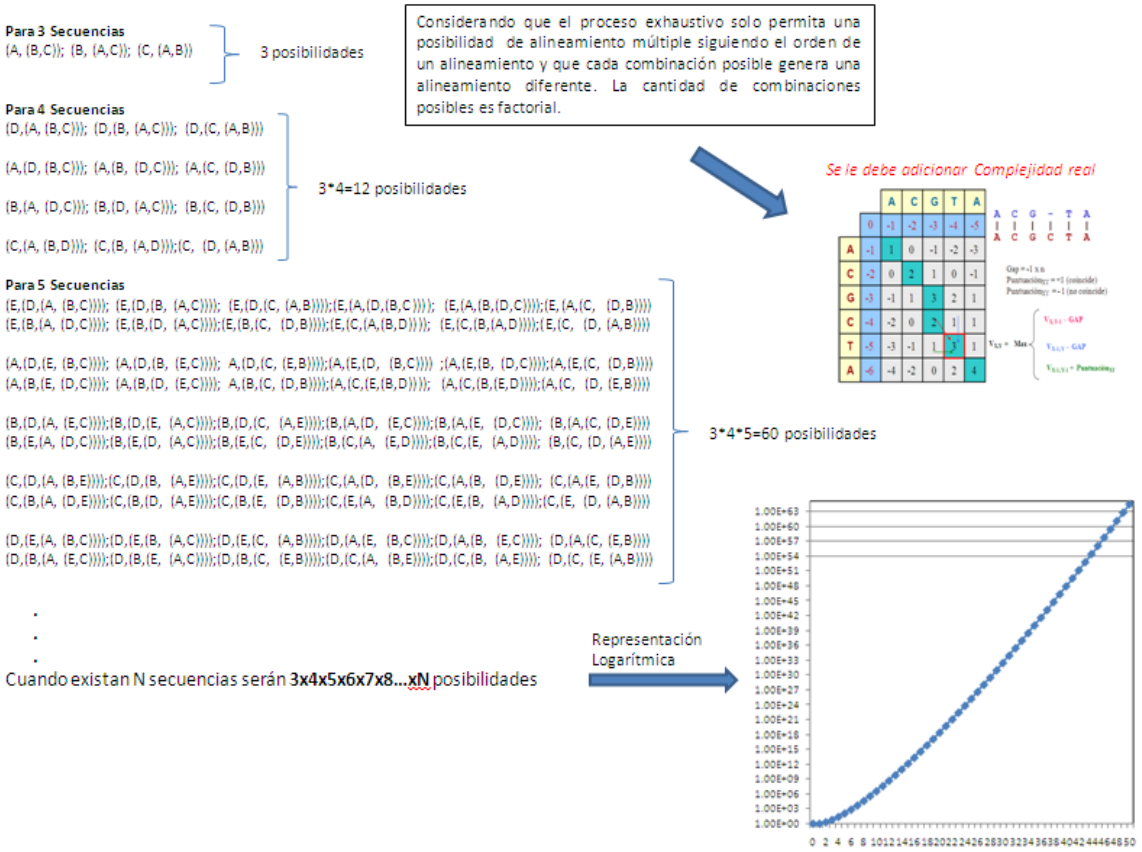


Figura I.7 Ejemplo de alineamiento exhaustivo de secuencias.

En el ejemplo de la Figura I.7 intentamos mostrar cuán complejo es hacer una búsqueda exhaustiva. Suponiendo la aproximación de que en un alineamiento múltiple se considerase una sola posibilidad de alineamiento para dos secuencias y que para explorar todos los posibles alineamientos se considerase solamente el orden completo de elección de las secuencias (*las combinaciones representadas en la figura para cantidades de secuencais diferentes*), se reduciría el problema a las combinaciones posibles representadas en la Figura I.7. Aún así, el comportamiento de posibilidades es factorial, lo que significa que la cantidad de posibles alineamientos de 50 secuencias es un valor exponencial de orden 64 (*gráfico representado en la* Figura I.7 *parte inferior*). Si a esto se le suma que cada posición de las secuencias a alinear se deben considerar respecto a todas, y no respecto a las que le preceden en el orden como se ha presentado en la aproximación del problema anterior, el valor del exponencial 64 crece mucho más y solo se habla de 50 secuencias.

Si intentamos representar gráficamente el alineamiento de 3 secuencias de la misma forma que se representa el algoritmo de programación dinámica para dos secuencias (Figura I.8), en el caso de tres secuencias sucede que el análisis de celdas, no es en una matriz sino en un cubo (*representar el cubo en un eje de coordenadas XYZ, donde las secuencias a alinear se disponen en cada uno de los ejes coordenados, o aristas ortogonales del cubo*), el esquema de puntuación para cada celda es más complejo, porque se deben tener en cuenta más posibilidades de alineamientos que en el de pares de secuencias. Una vez obtenidos todos los valores, se debe buscar el camino óptimo en el cubo (Figura I.8). Visto así, cada secuencia más a alinear en un alineamiento múltiple, equivale a aumentar una dimensión en el análisis (*generación de un espacio n-dimensional*), llevando este algoritmo a un orden de complejidad $O(L^N)$, donde L representar el valor medio de longitud de secuencia y N la cantidad de secuencias a alinear. Esta es la razón de la complejidad de los algoritmos de alineamiento múltiple de secuencias. El alineamiento de secuencias que contempla todas las posibilidades y la construcción del espacio *n-dimensional*, es clasificado como alineamiento exhaustivo.

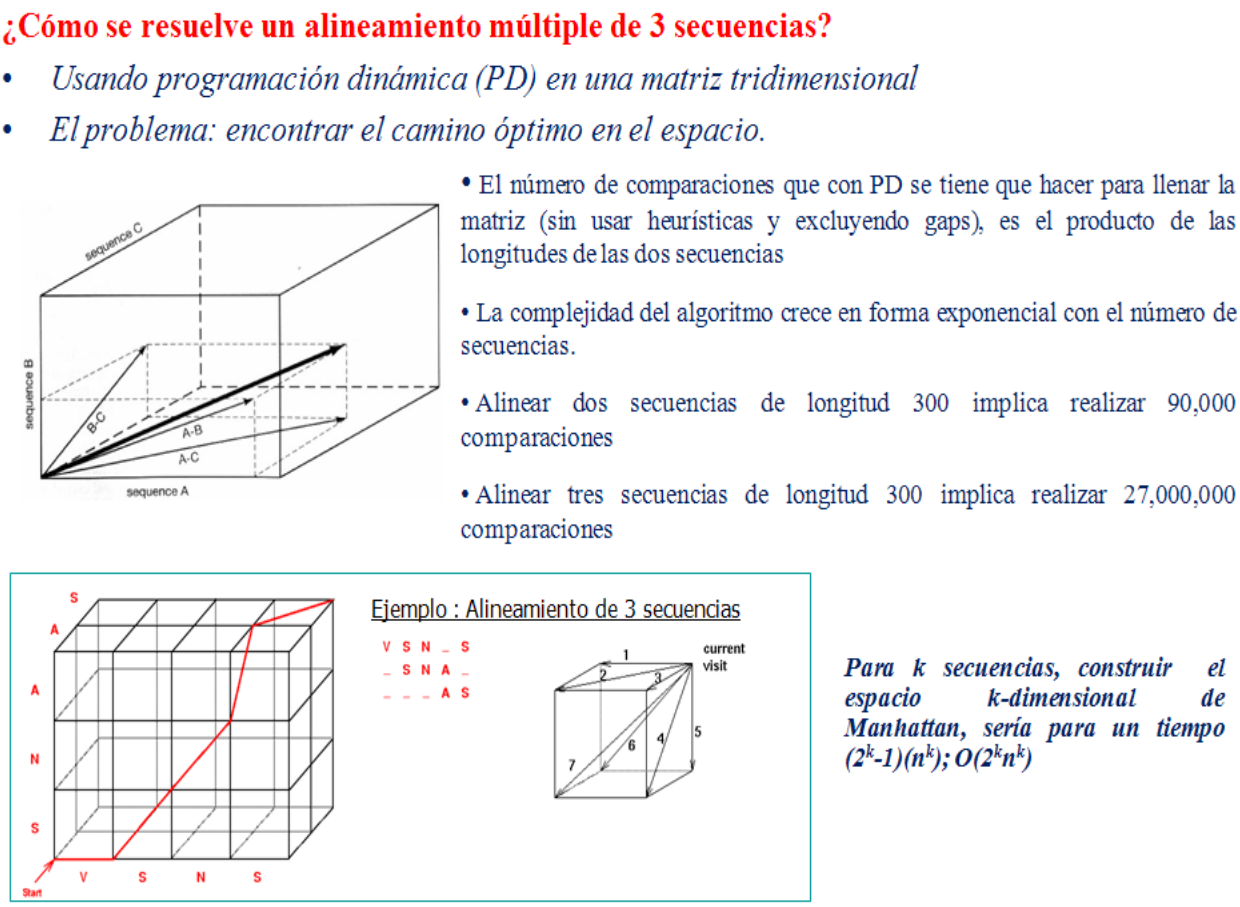


Figura I.8 Problema de alineamiento múltiple de secuencias.

I.3 Estado del Arte de algoritmos de alineamiento de secuencias

Teniendo en cuenta el consumo de recursos que supone la ejecución de aplicaciones de alineamiento múltiple de secuencias en los entornos de cómputo, se han destinado incansables esfuerzos en la búsqueda de estrategias que permitan aumentar el número de secuencias a tratar (*manteniendo buena calidad en los resultados*), clasificándose las estrategias de alineamiento múltiple de secuencias en: exhaustivas y heurísticas. Las heurísticas comprende un gran variedad de estrategias como las progresivas, iterativas, estructurales, consistency-based. Una organización esquemática de los algoritmos y las estrategias que emplean, se puede observar en la Figura I.9.

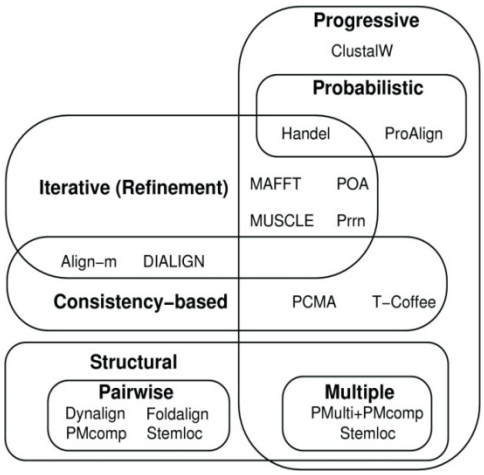


Figura I.9 Algoritmos de alineamiento múltiple de secuencias.

Los algoritmos de alineamiento múltiple de secuencias tienen en común, que a pesar de la estrategia que los caracterice, todos emplean los alineamientos en pares de secuencias, sean tanto globales como parciales. Dentro de las categorías de algoritmos de alineamiento múltiple de secuencias antes descritas, los algoritmos más utilizados usan heurísticas. En los últimos años los algoritmos que se han destacado por su exactitud son MUSCLE (Deng *et al.,* 2006), T-Coffee (Notredame *et al.,* 2000) y ClustalW (Rezaei & Monwar, 2006).

I.3.1 Algoritmos exhaustivos

Los algoritmos de alineamiento con estrategias exhaustivas analizan todas las posibles soluciones del espacio *n-dimensional* (*a través de programación dinámica, descrita anteriormente*). El modelo consume mucho tiempo de cómputo y memoria por el grado de complejidad exponencial dependiente del número de secuencias, solo pueden emplearse para un número reducido de secuencias a alinear.

I.3.2 Algoritmos heurísticos

Los algoritmos heurísticos para obtener una solución reducen las posibles soluciones del problema. La mayoría de los algoritmos de alineamiento múltiple de secuencia son heurísticos. Se diferencian por la forma en que intentan aumentar la exactitud del resultado final.

I.3.2.1 Algoritmos progresivos

Los algoritmos de esta categoría (*ClustalW (Rezaei & Monwar, 2006)*; *T-Coffee (Notredame et al., 2000))* comprenden varias etapas. Primero se obtienen los alineamientos en pares para todas

las secuencias. Posteriormente con los resultados de los alineamientos en pares se crea una matriz de distancia (*valores de similitud de secuencia*), para ver la relación de cada secuencia con las demás. Con la matriz de distancia se crea un árbol guía (*empleando métodos como el Neighbor-Join (Saito & Nei, 1987) o UPGMA (Unweighted Pair Group Method with Arithmetic mean; Sokal & Michener, 1958))* (Figura I.10).

En la Figura I.10 se muestra una representación de una matriz de distancia para cuatro secuencias y un árbol guía que es empleado para guiar el alineamiento de las cuatro secuencias. En la matriz de distancia cada celda representa la similitud que hay entre pares de secuencias (*en la matriz de la figura la celda D_{12} representa el valor de distancias de la secuencia S_1 con la S_2*). La distancia puede ser calculada bajo algún criterio, como por ejemplo la similitud de secuencia. Con la matriz de distancia se construye un árbol que no es más que una forma de relacionar las secuencias considerando algún criterio como cercanía. Supongamos que se desea crear el árbol juntando las secuencias más cercanas; se comienzan a agrupar las secuencias con mayor porcentaje de similitud, y luego se re-calcula la distancia en la matriz considerando la agrupación anterior, obteniendo una matriz más pequeña que vuelve a servir para encontrar las siguientes más cercanas.

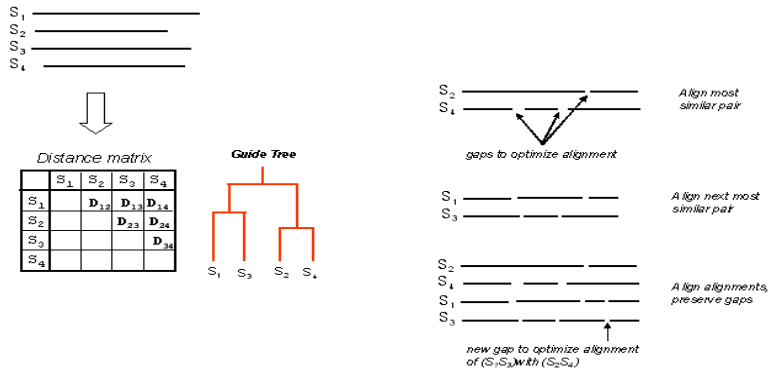


Figura I.10 Representación de matriz de distancia y árbol guía.

Una vez construido el árbol guía, las secuencias se alinean de forma progresiva siguiendo las ramas del árbol y uniéndose en grupos de dos secuencias al inicio y alineamientos intermedios posteriormente. El algoritmo tiene la ventaja de reducir el espacio de posibilidades a un solo alineamiento. Tiene la desventaja que como se define un único alineamiento en el árbol guía, cualquier error producido afecta el resultado final en gran medida.

I.3.2.2 Algoritmos iterativos

Los algoritmos iterativos intentan encontrar la solución óptima a un problema, modificando una solución sub-óptima (*MUSCLE (Deng et al., 2006)*). El proceso comienza con una solución de baja calidad, que es modificada de forma iterativa hasta que no haya mejora. El algoritmo consiste en la obtención de alineamientos en múltiples iteraciones, de dos formas: una se realiza dividiendo las secuencias de forma aleatoria y otra empleando un árbol guía obtenido por el método UPGMA (***U****nweighted* ***P****air* ***G****roup* ***M****ethod with* ***A****rithmetic mean*). Los resultados se agrupan para obtener un único alineamiento y se realiza de nuevo el proceso, pero utilizando el alineamiento obtenido como inicio. La operación se realiza hasta que la mejora en el resultado no sea significativa. Tienen la ventaja que refina el resultado en cada una de sus fases iterativas, pero tiene la desventaja que es muy dependiente de la solución inicial si no se incluye información biológica extra.

I.3.2.3 Algoritmos consistency-based

Esta clasificación de algoritmos utilizan información biológica (*T-Coffee (Notredame et al., 2000); MUSCLE (Deng et al., 2006)*) contenida en las secuencias de entrada, analizando los alineamientos en pares, la frecuencia de cambio de residuos, la relación entre residuos (*forma genérica de referirse a aminoácidos o nucleótidos*) y posibles regularidades. En la misma línea introducen información estructural de las secuencias de entrada, y emplean bases de datos biológicas. Tiene la ventaja que ayudan a mejorar los resultados porque tienen en cuenta la información que se encuentra de forma implícita en los datos e información conocida.

I.3.2.4 Algoritmos estructurales

Los algoritmos estructurales incluyen información estructural a los modelos de análisis de datos, contrastando las estructuras con las secuencias. Obtienen ventaja, dado que utilizan información conocida y presente en las bases de datos.

I.3.3 Limitantes del alineamiento de secuencias

A pesar de la variedad de algoritmos de alineamiento múltiple de secuencias que existen, el problema de alineamiento múltiple de secuencias aún no es un problema resuelto, debido a que se basan fundamentalmente en la implementación de heurísticas.

En los últimos años, multitud de investigaciones abordan este problema desde la computación de altas prestaciones, tratando de explotar los recursos de cómputo vigentes (Helal *et al.,* 2008; Saeed & Khokhar, 2008). La utilización de HPC, lenguajes paralelos de programación, entornos de cómputo paralelo y bases de datos segmentadas de acceso múltiple, es lo que muestran las investigaciones más recientes publicadas (Rezaei *et al.,* 2006; Rezaei & Monwar, 2006).

En la actualidad con la creación de nuevas técnicas experimentales y con la velocidad de secuenciación de genomas, se están produciendo grandes volúmenes de datos como se muestra en la Figura I.11. La cantidad de pares de bases en las secuencias publicadas en GenBank (*base de datos de información genética, http://www.ncbi.nlm.nih.gov/Genbank/GenbankSearch.html*) en la actualidad es mayor a los 90 000 millones. Este volumen de datos solo puede ser procesador en entornos de cómputo paralelo.

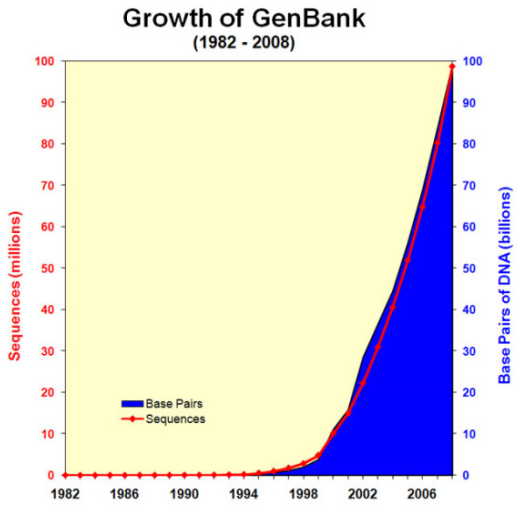


Figura I.11 Gráfica del crecimiento de las secuencias biológicas en el tiempo.

Estos datos necesitan procesamiento masivo (*alineamientos de miles de secuencias*). Este es un problema muy delicado, porque si los algoritmos actuales de alineamiento de secuencias, no han resuelto el problema normal de alineamiento múltiple, mucho menos se podrán enfrentar a tales volúmenes de datos. En otras porque las limitaciones actuales en los algoritmos se agravan cuando hay multitud de secuencias de entrada para analizar. Es por ello, que el problema de alineamiento múltiple de secuencias ha pasado a otra escala de necesidad de cómputo de altas prestaciones y cambio de estrategias en los algoritmos. Esta necesidad ha creado un vínculo cada vez más fuerte, entre la biología computacional y el HPC (HPC, d*el inglés* ***H****igh* ***P****erformance*

Computing), que podría nombrase como Bio-HPC. En otras palabras, el contexto para resolver el alineamiento múltiple de miles de secuencias es la computación de altas prestaciones.

Otro ejemplo claro de la complejidad que representa analizar grande volúmenes de datos, se pueden observar en la Figura I.12, donde se muestra la cantidad de árboles (*relaciones entre elementos, dependientes del orden en que se realicen*) que se pueden crear cuando crece el número de elementos a relacionar.

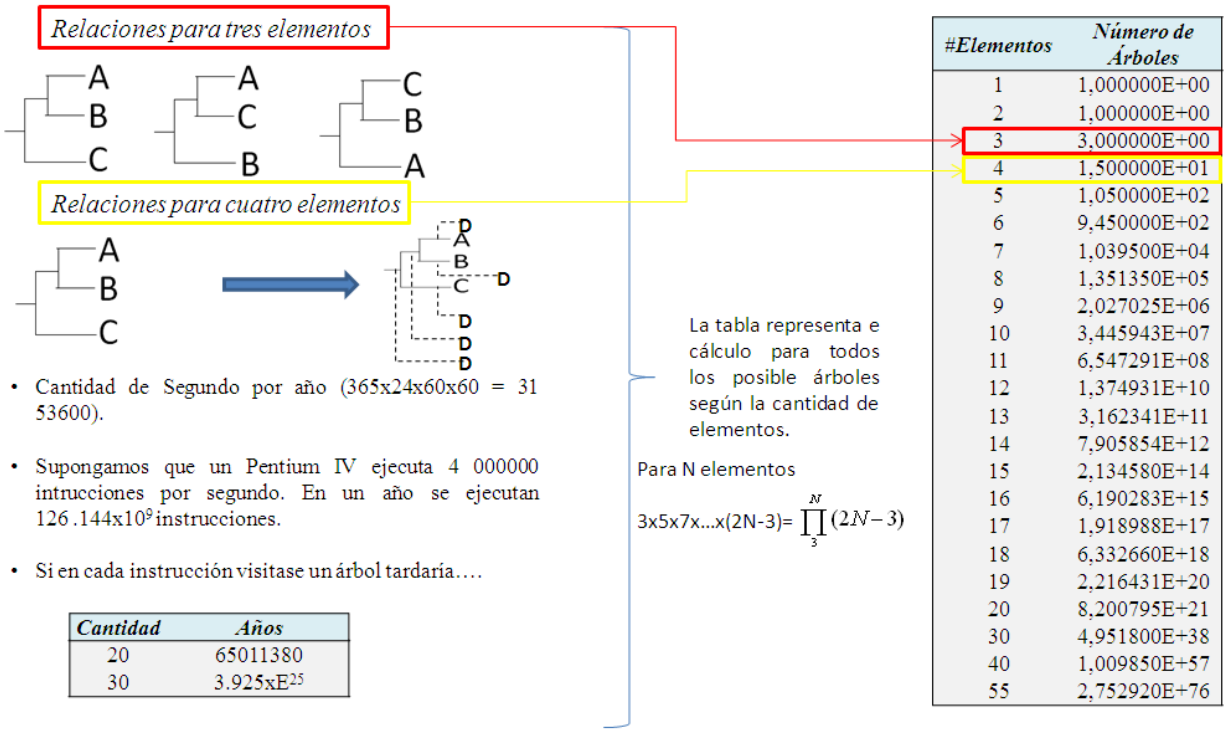


Figura I.12 Representación de lo que equivale calcular todas las posibles relaciones (*mediante árboles*) entre diferentes elementos.

En la parte superior de la figura se representan los tres posibles árboles para el alineamiento de 3 secuencias y debajo como empieza a aumentar con una secuencias más. El resultado se puede presentar con una expresión matemática (*productoria*) que aumenta muy velozmente con la cantidad de secuencias a alinear. La Tabla de la derecha en la figura muestra todos los posibles árboles a construir en dependencia de la cantidad de elementos, en este caso secuencias y además se pone un ejemplo de cuanto consumiría en años resolver todas las posibilidades para 20 y 30 secuencias en un solo ordenador. Este proceso de construcción de árboles es muy común en el alineamiento múltiple de secuencias, y se puede traducir en la cantidad de alineamientos

múltiples diferentes que se pueden resolver, y ese volumen de cómputo solo puede resolverse en computadores paralelos diseñados para cómputo masivo.

I.3.4 Algoritmo de Alineamiento múltiple T-Coffee

Entre los algoritmos de alineamiento múltiple de secuencias se encuentra T-Coffee (Figura I.13). Este algoritmo se destaca por la elevada exactitud en la obtención de un alineamiento múltiple para 100 secuencias. Fue desarrollado en el Centro de Regulación Genómica de Barcelona (*CRG*), centro con el que actualmente se colabora, y actualmente el algoritmo es empleado para el estudio de enfermedades como el cáncer, estudios de genomas y análisis evolutivos. Es de los algoritmos que por su aplicación tiene un desarrollo constante, existiendo un proyecto para su mantenimiento. T-Coffee está clasificado como algoritmo progresivo basado en consistencia (*introduce información biológica a partir de los datos de entrada*), y en la última década es el mejor de su categoría.

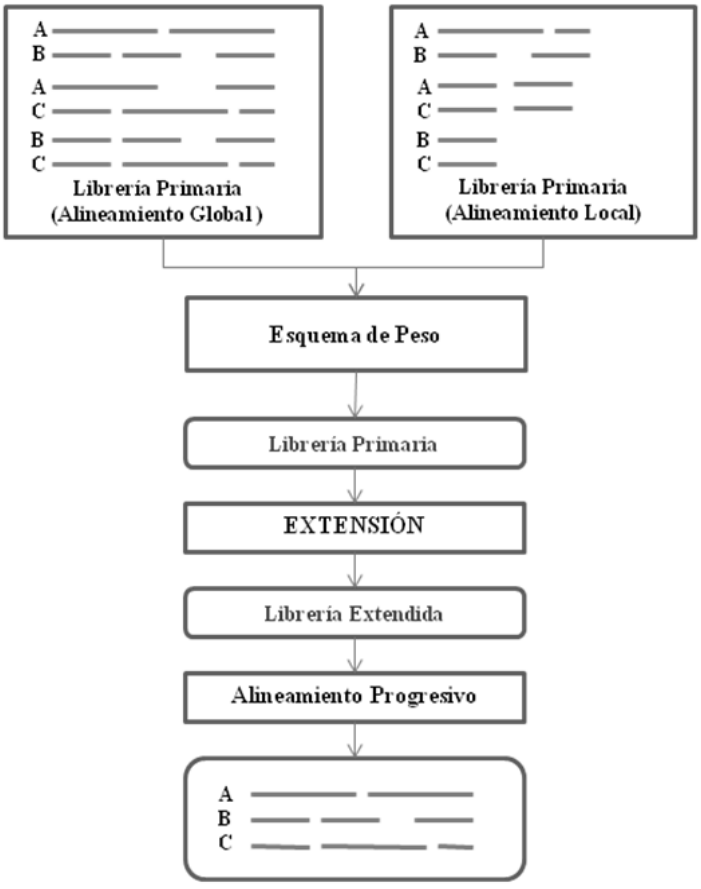


Figura I.13 Esquema de las fases del algoritmo de alineamiento múltiple T-Coffee.

El algoritmo T-Coffee está compuesto por tres fases de procesamiento de las secuencias de entrada para dar lugar al alineamiento múltiple, que son: una fase de construcción de una librería primaria, una de construcción de una librería extendida y la fase de alineamiento progresivo. T-Coffee intenta resolver los problemas del alineamiento progresivo introduciendo información de la secuencia, lo que implica un elevado grado de cómputo cuando el número de secuencias a alinear es del orden de miles. En el presente trabajo de investigación intentaremos resolver los problemas de los algoritmos de alineamiento múltiple, cuando intentan procesar miles de secuencias, mediante técnicas de paralelismo. Los entornos paralelos tanto para T-Coffee como para los algoritmos de alineamiento múltiple representan la única solución para analizar los elevados volúmenes de datos como los presentes en bases de datos genéticas y de proteínas, teniendo en cuenta además de la cantidad de posibles soluciones que representa analizarlas según las relaciones que pueden obtenerse por la construcción de árboles guía diferentes. A continuación se describirán algunas características de los entornos donde se puede dar solución al alineamiento múltiple de secuencias. En el caso de nuestra investigación las soluciones propuestas se enmarcaran dentro de los entornos de cluster de computadoras.

I.4 Contexto

I.4.1 Computación de Altas Prestaciones

Dentro de la ciencia computacional, la computación de altas prestaciones, es el campo en el que se estudia el conjunto de técnicas necesarias para obtener mejores prestaciones de los recursos de cómputo existentes. La HPC se apoya en supercomputadores y clusters de computadores, para dar solución a problemas de alta demanda de cómputo y tiempo como el descrito anteriormente de alineamiento múltiple de secuencias.

Las arquitectas para HPC (Hwang K, 1993) cada vez son más potentes, pero también mucho más complejas. En la búsqueda de arquitecturas cada vez más rápidas para dar respuesta al cálculo intensivo de las aplicaciones emergentes, se ha llegado a los computadores paralelos. Estos computadores se pueden definir, como sistemas de cómputo con gran capacidad de cálculo y procesadores de alto rendimiento para trabajar en paralelo.

La base de los computadores de alta demanda es la replicación de unidades funcionales, que pueden ser procesadores y/o dispositivos de memoria, aumentando así el nivel de cómputo y

manteniendo un costo computacional relativamente aceptable. Este tipo de arquitectura incluye una conexión entre las unidades funcionales mediante una red de interconexión. Así se pueden ejecutar de forma conjunta tareas de cómputo concurrentes. Una organización taxonómica de estos sistemas de cómputo se puede observar en la Figura I.14.

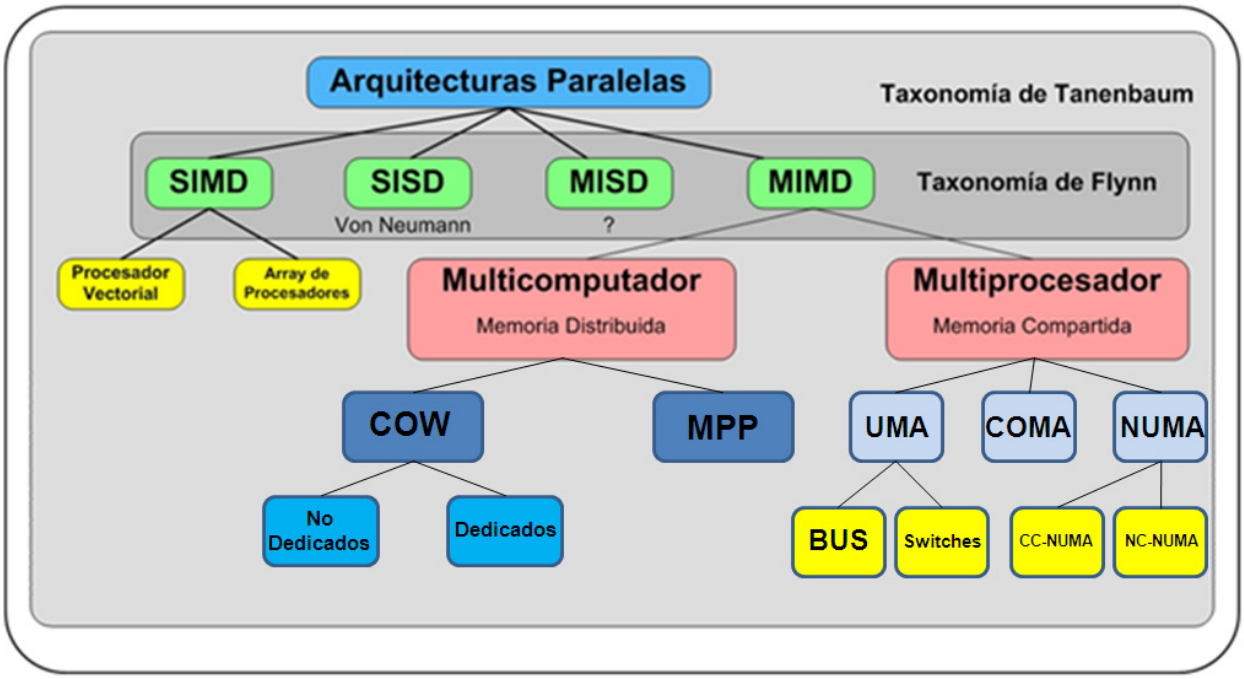


Figura I.14 Taxonomía de Tanenbaum para arquitecturas paralelas.

Los computadores paralelos en la taxonomía de Tanenbaum, según la categorización de Flynn (Flynn, 1972), se pueden clasificar en dependencia del flujo de instrucciones y el flujo de datos en cuatro categorías de paralelismo (Figura I.15):

- ✓ Computadores ***SISD*** (*del inglés **S**ingle **I**nstruction, **S**ingle **D**ata*). Son los computadores con una única unidad de procesamiento. El procesador es capaz de realizar acciones de forma secuencial, controladas por un programa que se encuentra almacenado en una memoria conectada al procesador. Este hardware está diseñado para dar soporte al procesamiento secuencial clásico, basado en el intercambio de datos entre la memoria y los registros del procesador, así como la realización de operaciones aritméticas entre ellos.
- ✓ Computadores ***SIMD*** (*del inglés **S**ingle **I**nstruction, **M**ultiple **D**ata*). Son computadores paralelos con un flujo único de instrucciones que opera sobre múltiples datos simultáneamente (paralelismo de datos). En esta clasificación un único programa controla los

procesadores. Es útil en aplicaciones uniformes, como el procesamiento de imágenes y multimedia.

- ✓ Computadores ***MISD*** (del inglés **M**ultiple **I**nstruction, **S**ingle **D**ata). Es una clasificación de computadores que hasta el momento es conceptual.
- ✓ Computadores ***MIMD*** (*del inglés **M**ultiple **I**nstruction, **M**ultiple **D**ata*). Son computadores con varios flujos de instrucciones y de datos, realizándose simultáneamente en paralelo. Tienen la característica de poder trabajar asíncronamente.

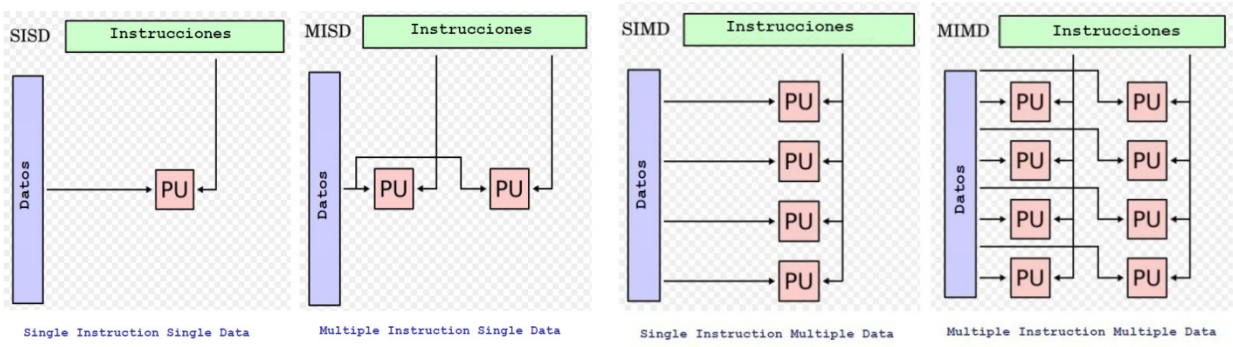


Figura I.15 Categorización de Flynn para computadores paralelos.

I.4.1.1 Computadores MIMD

Los computadores MIND pueden clasificarse en dos tipos de acuerdo al mecanismo empleado para comunicación y sincronización: computadores de memoria compartida o computadores de memoria distribuida.

Computadores MIMD de memoria compartida

Los computadores de memoria compartida (Figura I.16) o computadores multiprocesadores (Culler & Singh, 1999; Wilkinson, 1996) están compuestos por procesadores autónomos que trabajan asíncronamente. La comunicación entre ellos se realiza a través del recurso compartido (*un espacio de direcciones global accesible a todos los procesadores*). El proceso de comunicación y sincronización se realiza de forma explícita, de manera que el emisor escribe y el receptor lee de la memoria global. Tienen la limitante de que un fallo de memoria de algún componente puede causar un fallo total del sistema. Además el incremento del número de procesadores puede llevar a problemas de acceso a memoria.

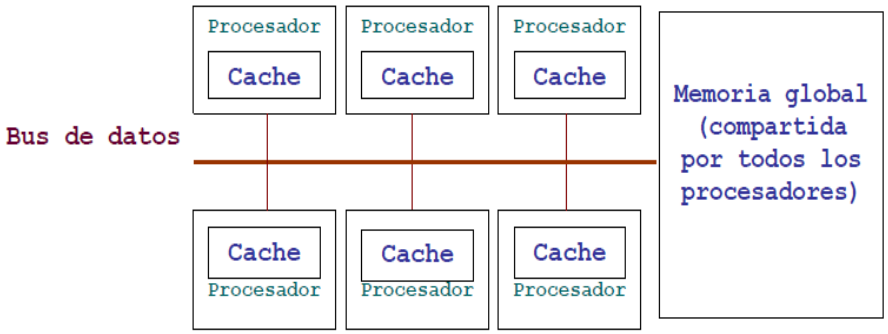


Figura I.16 Arquitectura MIMD con memoria compartida.

En estos computadores, el bus de comunicaciones es un cuello de botella que limita la escalabilidad a un máximo de pocas decenas de procesadores.

Los mecanismos físicos para compartir los datos que emplean los computadores de memoria compartida son:

- ✓ **UMA** (*del inglés **U**niform **M**emory **A**ccess*): Donde todos los procesadores acceden a la memoria de forma uniforme en el mismo tiempo.
- ✓ **NUMA** (*del inglés **N**on-**U**niform **M**emory **A**ccess*). Formado por una colección de memorias separadas que forman un espacio de memoria direccionable.

Acceso uniforme UMA en Computadores MIMD de memoria compartida

En las arquitecturas MIMD con acceso uniforme a memoria (Figura I.17), toda la memoria global es accedida a través de una interconexión común. Por tanto, en ausencia de algún conflicto, el tiempo de acceso a cualquier posición de memoria compartida es uniforme. Esto es posible porque todas las memorias se encuentran a la misma distancia. Este tipo de sistemas es conocido como multiprocesadores simétricos (*SMP del inglés **S**ymetric **M**ulti **P**rocessor*).

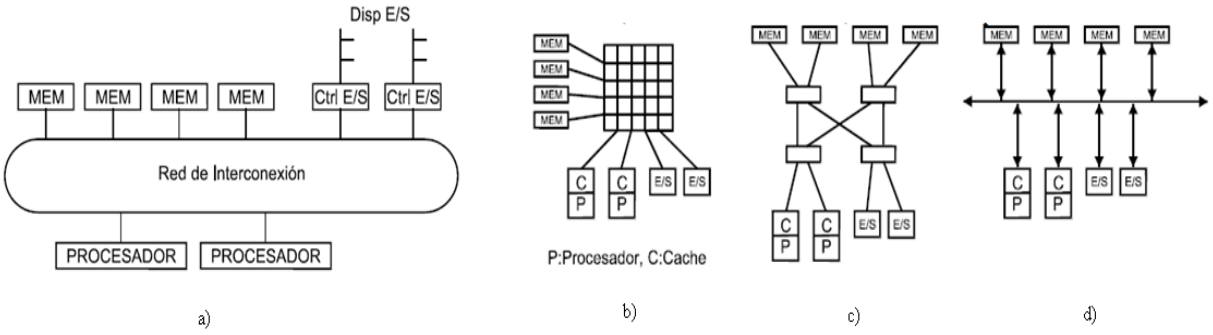


Figura I.17 Arquitecturas de multiprocesadores de memoria compartida de acceso uniforme. El caso a) representa la estructura básica, en la que todos las memorias conforman un espacio único y cualquier procesador puede acceder a cualquier posición de memoria. Los casos b), c) y d) muestran los esquemas típicos de interconexión, como el "*crossbar*", la red multi-etapa y el bus único, respectivamente.

Acceso no uniforme NUMA en Computadores MIMD de memoria compartida

En las arquitecturas MIMD con acceso no uniforme a memoria (Figura I.18), la memoria se encuentra distribuida entre los procesadores, de forma tal que cada procesador tiene una parte del total de la memoria compartida del sistema. Sin embargo existe un espacio de direccionamiento único, por lo que a pesar de que la memoria se encuentre físicamente distribuida, se encuentra lógicamente compartida entre todos los procesadores del sistema. A este tipo de computadores suele llamársele como computadores de memoria compartida distribuida (*DSM del inglés **D**istributed **S**hared **M**emory*).

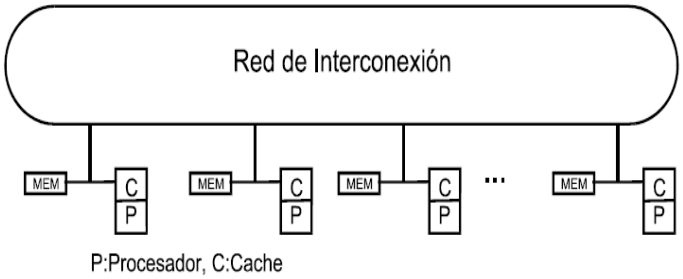


Figura I.18 Arquitectura de multiprocesador de memoria compartida de acceso no uniforme.

Computadores MIMD de memoria distribuida

Los computadores MIMD de memoria distribuida (Figura I.19), son sistemas en que cada procesador tiene su memoria local y solo pueden compartir esta información a través del paso de mensajes (Seitz, 1990; Athas & Seitz, 1998). Por lo que requieren un proceso de comunicación y sincronización a través de la red de interconexión. En este tipo de sistemas, la comunicación puede ser el cuello de botella e influye la topología de la red de interconexión. Es una arquitectura escalable para aplicaciones que sean apropiadas al diseño propio de la arquitectura. Además de ser fácilmente escalable, tiene la ventaja de poseer una alta disponibilidad (*el fallo de un CPU individual no tiene porqué afectar todo el sistema*).

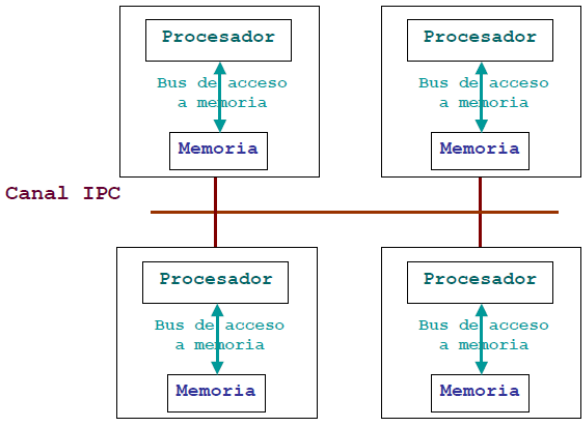


Figura I.19 Arquitectura MIMD con memoria distribuida.

Por lo general este tipo de computadores es programado mediante lenguajes con paralelismo explícito o mediante librerías para el paso de mensajes explícito, como PVM (*del inglés **P**arallel **V**irtual **M**achine*; Geist et al., 1994) o MPI (*del inglés **M**essage **P**assing **I**nterface*).

Las aplicaciones de la computación de altas prestaciones y los entornos de cómputo paralelo se extienden prácticamente a todos los ámbitos de análisis de datos científicos. Las posibilidades de cómputo, acceso a memoria y comunicación que brindan son necesarias en el problema de alineamiento múltiple de secuencias, donde el cómputo paralelo se vuelve necesario y parece ser la única solución frente al volumen de millones de datos a procesar.

I.5 Objetivos

Hasta este momento, hemos descrito el problema de las aplicaciones con carácter biológico, específicamente las de alineamiento múltiple de secuencias, y hemos presentado algunas de las características del entorno en el que puede ser solucionado el problema. Con estos elementos, hemos centrado nuestro interés en crear un modelo paralelo para el alineamiento múltiple de secuencias biológicas. Para la creación del modelo hemos decidido emplear como caso de estudio el algoritmo T-Coffee, por ser de los algoritmos que obtiene mayor exactitud en resultados de alineamientos para 100 secuencias; además porque la base del algoritmo permite extraer información biológica de los secuencias de entrada.

El modelo que propondremos debe tener las características de poder escalar el número de entrada de los algoritmos de alineamiento múltiple a miles, con lo que esto conlleva (*disminuir el consumo de recursos de cómputo y reducción el tiempo de ejecución*), manteniendo la calidad de los resultados como premisa fundamental.

Nos esforzaremos en la comprensión del modelo de alineamiento múltiple de secuencias, para lograr un mejor acoplamiento de la arquitectura del computador al programa o código que se ejecute en él, considerando las arquitecturas enmarcadas en el HPC.

Basado en los elementos antes expuestos, nos propusimos la siguiente ***hipótesis*** del trabajo: **Se puede crear un modelo para paralelizar los algoritmos de alineamiento múltiple de secuencias, que permita aumentar el número de datos de entrada obteniendo la misma calidad en los resultados, y disminuyendo a la vez el tiempo de cómputo y la cantidad de memoria a utilizar, si se logran: identificar las zonas en el algoritmo de las que depende la calidad del resultado y clusterizar la entrada de datos basado en el conocimiento implícito en los mismo, para que sea más preciso el análisis por el algoritmo.**

Para poder aceptar o refutar la hipótesis nos propusimos como ***objetivo general***:

- ✓ Crear un modelo para paralelizar los algoritmos de alineamiento múltiple de secuencias, que permita aumentar el número de secuencias de entrada obteniendo la misma calidad en los

resultados y disminuyendo a la vez el tiempo de cómputo, empleando como caso de estudio T-Coffee.

Y como ***objetivos específicos:***

- ✓ Estudiar los algoritmos de alineamiento de pares secuencias y alineamiento múltiple de secuencias.
- ✓ Estudiar las estrategias empleadas en estos algoritmos para emplear cómputo paralelo y distribuido en su optimización.
- ✓ Analizar las fases de procesamiento de datos del algoritmo de estudio T-Coffee.
- ✓ Proponer e implementar un modelo paralelo para el caso de estudio T-Coffee, que permita escalar el número de secuencias de entrada del algoritmo, manteniendo la calidad de los resultados.
- ✓ Hacer análisis de rendimiento del modelo propuesto.
- ✓ Validar el modelo, analizando la calidad de los resultados obtenidos por el modelo contra a T-Coffee.

La ***aportación teórica y novedad científica*** de este trabajo de investigación se enmarcan en dos aspectos. Primero, en el modelo propuesto para paralelizar aplicaciones de alineamiento de secuencias (manteniendo sus mismos principios de manejo de la información) para segmentar el problema de entrada en tareas a efectos de su ejecución independiente en sistemas de cómputo paralelo que al final contribuyen a un único resultado; y segundo, en las técnicas que permiten escalar el algoritmo a miles de secuencias, manteniendo la misma calidad en los resultados obtenidos. Todo este trabajo se ha realizado tomando como base al algoritmo T-Coffee, como caso de estudio.

El ***aporte práctico***, está en la implementación del modelo paralelo en el algoritmo T-Coffee. Esto ha permitido escalar el número de secuencias de entradas a miles, y reducir a la vez el tiempo de cómputo. Todo lo anterior sin alterar la calidad de los resultados obtenidos por el algoritmo. El modelo propuesto es un modelo paralelo e independiente del algoritmo de alineamiento múltiple, lo que permite ser empleado en algoritmos con estrategias diferentes a la de T-Coffee. Además por haber empleado como plataforma T-Coffee, si se logran pasar las pruebas de calidad, la aportación podría formar parte de T-Coffee y estaría disponible a toda la comunidad científica.

Estructura de la Memoria

En el **actual capítulo** se presentó una visión general del problema de las aplicaciones de alineamiento múltiple de secuencias Se introdujeron algunos conceptos biológicos necesarios para comprender el problema. Se explicó en qué consiste y lo que significa el alineamiento de secuencias biológicas. Se mostró que la única vía para solucionar el problema es la aplicación de la Computación de Altas Prestaciones y finalmente se presentaron los objetivos de la investigación y las aportaciones que supone.

En el **capítulo II** se presentará la propuesta del modelo paralelo para los algoritmos de alineamiento múltiple de secuencias. Como se utiliza T-Coffee como algoritmo de referencia, se describen las fases de T-Coffee, se analizan y posteriormente se propone el modelo paralelo. A continuación se detallan las fases del modelo y se define la forma de validación. El modelo paralelo que se propone en el capítulo II tiene como objetivo aumentar la cantidad de secuencias de entrada del algoritmo de alineamiento de secuencias T-Coffee a miles de secuencias, manteniendo la calidad de los resultados, disminuyendo el tiempo y el consumo de recurso de cómputo.

El **capítulo III** presenta y discute una serie de experimentos para poder validar el modelo paralelo propuesto para mejorar los algoritmos de alineamiento múltiple de secuencias tomando como plataforma de prueba el algoritmo T-Coffee. Inicialmente se estudia el comportamiento de T-Coffee a través de un grupo de experimentos de rendimiento, posteriormente se muestran los resultados obtenidos por el modelo desde el punto de vista biológico para verificar la calidad de los resultados, y por último se presentan los resultados de rendimiento para el modelo propuesto a través de un grupo de experimentos que se realizan.

El **capítulo IV** presenta las conclusiones de la investigación desarrollada y las líneas abiertas que quedan en el problema de alineamiento múltiple de secuencias.

El **capítulo V** contempla la bibliografía utilizada para poder llevar a cabo toda la investigación.

CAPITULO II

Modelo paralelo de alineamiento múltiple de secuencias

En este capítulo se presentará el modelo paralelo propuesto para los algoritmos de alineamiento múltiple de secuencias. Debido a que se emplea T-Coffee como plataforma de análisis, se hará una introducción de T-Coffee, se describirán las fases de procesamiento de secuencias del algoritmo y posteriormente se pasarán a analizar las fases para proponer el modelo paralelo. A continuación se detallan las fases del modelo y se define la forma de validación. El objetivo del modelo paralelo que se propone en el capítulo es aumentar la cantidad de secuencias de entrada del algoritmo de alineamiento de secuencias T-Coffee a miles de secuencias, manteniendo la calidad de los resultados, disminuyendo el tiempo y el consumo de recurso de cómputo.

II.1 Introducción al algoritmo de alineamiento múltiple T-Coffee

T-Coffee es un algoritmo de alineamiento múltiple de secuencias que se destaca por la exactitud en la obtención de resultados para 100 secuencias. Fue desarrollado en el Centro de Regulación Genómica de Barcelona (*CRG*). T-Coffee está clasificado como algoritmo progresivo consistency-based y en la última década es el mejor de los métodos progresivos.

II.1.1 Definición de T-Coffee

T-Coffee (*del inglés **T**ree-based **C**onsistency **O**bjective **F**unction for alignment **E**valuation*) es un algoritmo progresivo de alineamiento múltiple de secuencias basado en consistencia (Notredame *et al.,* 2000). Este algoritmo intenta minimizar los errores introducidos por el árbol guía (*explicado anteriormente en el Acápite I.3.2.1 de algoritmos progresivos*) en el alineamiento progresivo, teniendo en cuenta información biológica contenida en las secuencias de entrada y extraída de una serie de alineamientos globales y locales en pares.

Los rasgos principales de T-Coffee son: la construcción de una librería de pares de alineamientos globales y locales, y un método heurístico que es usado para encontrar el alineamiento múltiple

que mejor ajusta los alineamientos en pares de la librería construida. Estas características son las que le proporcionan robustez al algoritmo y le permite considerar información de todas las secuencias de entrada.

En el algoritmo T-Coffee se pueden identificar tres fases (Figura II.1): dos primeras de construcción de una librería de alineamientos, nombrada al inicio *librería primaria* (*alineamientos globales y locales*), y posteriormente nombrada *librería extendida* (*donde se introduce además información de la coincidencia de pares de residuos (aminoácidos), de dos secuencias a través de una tercera*), y una última de construcción del *alineamiento múltiple progresivo*.

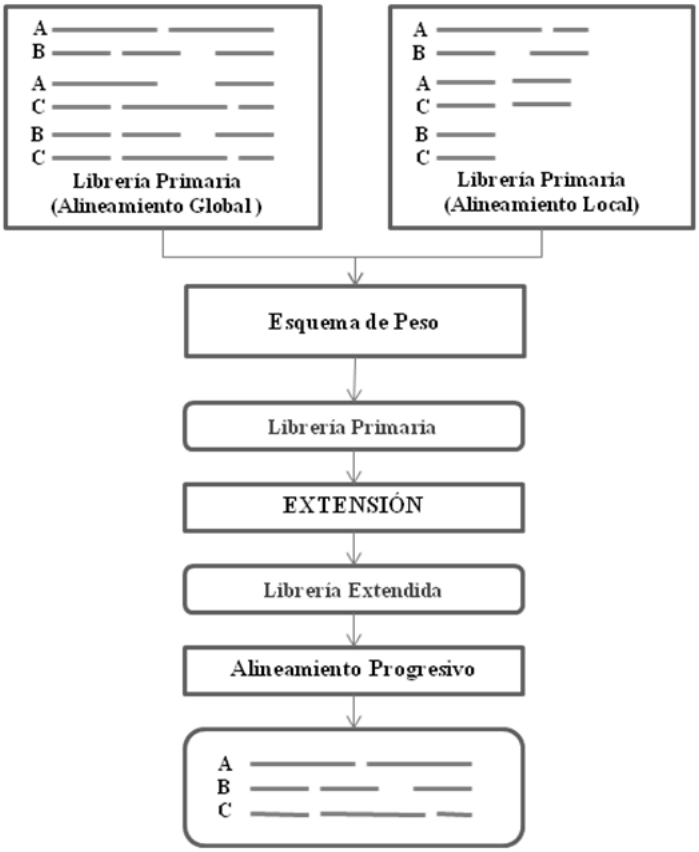


Figura II. 1 Representación de las fases de T-Coffee

Una de las características de T-Coffee es que consume un número elevado de recursos de cómputo (tiempo de cómputo, y memoria), por la forma en que está implementado el algoritmo debido a esta necesidad de construir librerías de alineamientos para cada secuencia (*más conocimiento en el algoritmo produce más cómputo*), limitando así la capacidad de procesar a

partir de un cierto número de secuencias de entrada un tiempo razonable. Es un algoritmo lento para 100 secuencias, pero no para valores muy superior a este.

II.1.2 Librería primaria

La fase de construcción de la librería primaria en T-Coffee, comprende el alineamiento en pares para todas las secuencias de entrada. Esta fase puede incluir más de un alineamiento para el mismo par de secuencias (*puede existir redundancia en la información*). En la librería primaria se obtiene información para N(N-1)/2 pares de secuencias, incluyendo datos de alineamientos globales y locales. En la Figura II.2, se muestra un ejemplo de alineamientos para cuatro frases sobre GARFIELD en la librería primaria, y los valores de peso asignado a cada alineamiento (*porcentaje de similitud de las secuencias alineadas*). Si se presta atención en el alineamiento mostrado en la parte superior derecha, se ve que se ha alineado CAT con FAT por un error en el alineamiento, producido por el orden en que se escogen las secuencias en el alineamiento progresivo, por este motivo T-Coffee introduce la librería primaria.

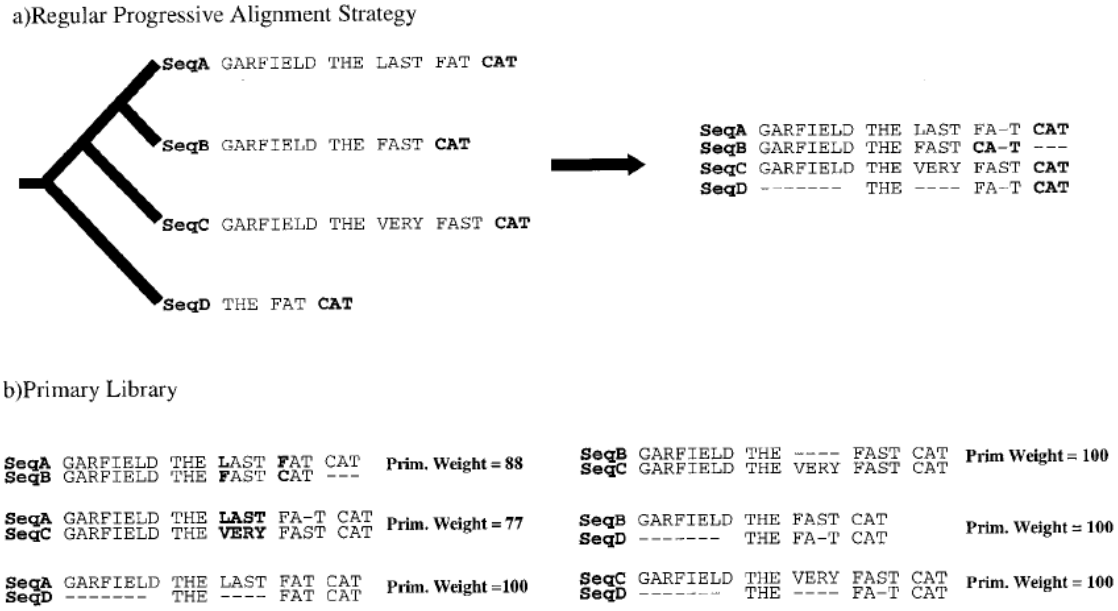


Figura II.2 Ejemplo de alineamientos obtenidos para construir la librería primaria.

En la librería primaria, cada alineamiento es representado como una lista de correspondencia de pares de residuos (***ej***. *el residuo* ***x*** *de la secuencia* ***A*** *es alineado con el residuo* ***y*** *de la secuencia* ***B***). Cada una de estas correspondencias es considerada por T-Coffee como una restricción, a la que luego se le asigna un peso, a partir de un esquema de puntuación.

El costo computacional de esta fase es muy elevado, porque se deben obtener los alineamientos en pares para todas las secuencias de entrada. El alineamiento en par de dos secuencias tiene una complejidad computacional de $O(L)$ en espacio y $O(L^2)$ en tiempo, donde L es la longitud media de las secuencias a alinear. Alinear N secuencias, requiere N(N-1)/2 combinaciones que equivale a $O(N^2)$, lo que resulta en una complejidad $O(N^2L^2)$ en tiempo y $O(N^2+L)$ en espacio. Esta fase cuando se desean alinear más de 100 de secuencias, ralentiza mucho al algoritmo T-Coffee, siendo un punto crítico de análisis para mejorar el rendimiento del algoritmo.

Esquema de pesos en la librería primaria

El esquema de puntuación de T-Coffee para asignar un peso a las restricciones de la librería primaria, emplea el valor de identidad de secuencia obtenida de los alineamientos en pares. En este esquema, se le asigna a cada par de residuos alineados un valor igual, al porcentaje de identidad del alineamiento de origen (Figura II. 2). Una de las ventajas de este esquema es la simplicidad. Además el esquema de pesos le proporciona a la función de puntuación de T-Coffee que devuelve el *score* (*valor que representa la calidad de alineamiento,* Ecuación II.1), una elevada efectividad. Esta función de score contempla el conocimiento en las secuencias de entrada por el análisis de los alineamientos en pares.

$$\textbf{COFFEE score} = \frac{\left[\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} W_{ij} \times SCORE\left(A_{ij}\right)\right]}{\left[\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} W_{ij} \times LEN\left(A_{ij}\right)\right]}$$

N → número de secuencias,

$A_{i,j}$ → proyección del par alineado de las secuencias S_i y S_j , obtenido del alineamiento múltiple,

$LEN(A_{i,j})$ → longitud de $A_{i,j}$,

$SCORE(A_{i,j})$ → identidad de secuencia de $A_{i,j}$ y alineamiento en par correspondiente en la librería,

$W_{i,j}$ → peso asociado al alineamiento en par de las secuencias S_i y S_j

Ecuación II.1 Ecuación de puntuación de T-Coffee.

El objetivo final de la librería primaria es lograr una combinación eficiente de la información de los alineamientos globales y locales. Esta operación se realiza después de la asignación de pesos a cada par de residuos. La combinación se realiza analizando cada correspondencia entre pares de residuos. Se buscan todas las incidencias de cada correspondencia entre pares de residuos anteriormente puntuado y se suman los valores de peso a la correspondencia en los alineamientos

(*se refuerza el valor de la correspondencia de residuos según la aparición en los alineamientos globales y locales*). La correspondencia entre pares de residuos que no son observados en la librería, no es representada.

El enorme incremento en los valores de peso de pares de residuos en la librería primaría, hace que se deba utilizar otro valor de peso menor (*obtenido a partir de este análisis*), que refleje la consistencia de cada par de residuos en las secuencias de entrada. Esta operación se realiza en el proceso que le precede, que es una extensión de la librería y es la segunda fase del algoritmo.

II.1.3 Librería extendida

El proceso de ajustar el grupo de restricciones de peso al alineamiento múltiple, es un problema conocido como problema NP-complete (Kececioglu, 1993), T-Coffee esquiva este problema con una heurística, que es la generación de la llamada librería extendida (Figura II.3).

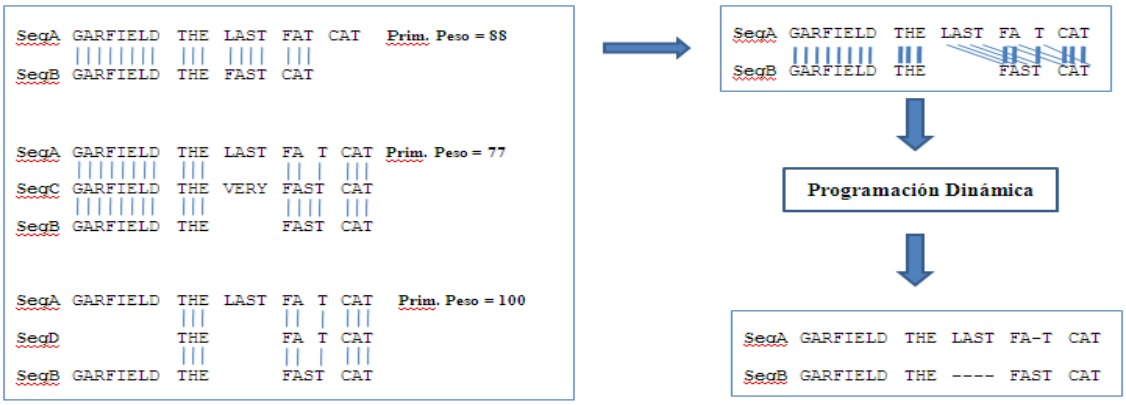


Figura II.3 Análisis en la librería extendida para la SeqA y SeqB

El análisis se basa en tomar de la librería primaria cada par de residuos alineados y chequear estos a través de todas las demás secuencias. Por ejemplo, supongamos que tenemos las secuencias ***A***, ***B***, ***C*** y ***D***, que se muestran en la Figura II.2, designemos ***A(G)*** al residuo *G* de GARFIELD en la secuencia ***A***, ***B(G)*** al residuo *G* correspondiente en la secuencia ***B*** y ***W(A(G), B(G))*** al peso asociado a este par de símbolos en la librería primaria. En el alineamiento directo de la secuencia ***A*** y la secuencia ***B*** en la librería primaría, existe la correspondencia de G en GARFIELD y se le asigna el peso de 88 (*porcentaje de identidad del alineamiento*).

Ahora analicemos los residuos de las secuencias ***A*** y ***B*** a través de la secuencia ***C***. En la Figura II.4 se observa que ***A(G)*** y ***C(G)*** están alineados, al igual que ***C(G)*** y ***B(G)***, de esta forma

se ha obtenido un alineamiento de ***A(G)*** y ***B(G)*** a través de la secuencia ***C***. A esta correspondencia se le asocia un peso igual, al valor mínimo de peso entre W_1=(***A(G), C(G)***)=77 y W_2=(***C(G), B(G)***)=100, obtenidos en la librería primaria (Figura II.2). De esta forma el peso asociado al análisis del alineamiento del residuo ***A(G)*** y ***B(G)*** a través de la secuencia ***C*** es 77. Este nuevo valor es sumado al obtenido en la librería primaria para el par de residuo ***A(G)*** y ***B(G)*** en su alineamiento, siendo el resultado 165 (88+77). Esta operación se realiza para cada par de secuencias a través de las secuencias restantes, aunque en ocasiones las secuencias que forme parte del triplete (*análisis de dos secuencias a través de la tercera*) no aporte nada de información.

En resumen el peso asociado a cada par de residuos será igual a la suma de los pesos obtenidos por el análisis de tripletes, sobre todos las secuencias. Este proceso favorece el alineamiento progresivo, porque incluye información del comportamiento de los pares de residuos alineados en todas las secuencias. Tiene la desventaja que el costo computacional es elevado, siendo $O(N^3L^2)$ en tiempo y $O(N^3+L)$ en espacio. Esta fase es la que en mayor cuantía perjudica al método T-Coffee y es muy sensible a análisis para la mejora de rendimiento de la aplicación.

II.1.4 Alineamiento progresivo

Para obtener el alineamiento progresivo, primero se obtiene la matriz de distancia de las secuencias de entrada, y se construye el árbol guía que es el encargado del orden en que se agrupan las secuencias en el alineamiento progresivo. El algoritmo empleado para realizar esta tarea es *Neighbor-Join* (Saito & Nei, 1987) ó *UPGMA* (*del inglés **U**nweighted **P**air **G**roup **M**ethod with **A**rithmetic mean*; Sokal & Michener, 1958), en dependencia de la configuración definida a priori. Una vez obtenido el árbol guía, este es empleado para implantar el orden en el alineamiento múltiple progresivo como se muestra en la Figura II.4. Durante esta fase, se emplea la información de la librería extendida (*correspondencia entre pares de residuos y reflejada en los valores de peso*).

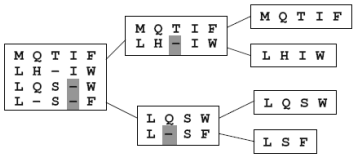


Figura II.4 Alineamiento progresivo ordenado por un árbol guía.

El alineamiento progresivo es construido tomando como esquema de puntuación la información de correspondencia entre residuos obtenida en la librería extendida. Esta fase no es muy costosa computacionalmente si se compara con las anteriores, pero es donde se decide el orden del alineamiento y por ende la calidad del resultado final.

II.2 Validación de algoritmos de alineamiento múltiple progresivo

La forma de evaluar la calidad de los algoritmos de alineamiento múltiple de secuencia es mediante *benchmarks* (Thompson *et al.,* 2005*;* Finn *et al.,* 2008). En el caso de *benchmarks* para algoritmos de alineamientos, están formados por bases de datos de secuencias de proteínas y ADN, que tienen además de ficheros con grupos de secuencias, los resultados de los alineamientos de estos grupos de secuencias. La información que brindan los *benchmarks* de esta categoría en muchos casos ha sido corroborada de forma experimental, de forma que se tiene una entrada para el algoritmo y un resultado de salida con el cual se puede comparar. Uno de los *benchmarks* más usados es BALIBASE (*del inglés Benchmark ALIgnment dataBASE;* Anexo I), que actualmente se encuentra en la versión 3.0. Esta base de datos tiene la característica de cubrir un espectro de categorías de familias de proteínas bastante amplio. En nuestra investigación como se propondrán cambios al algoritmos T-Coffee, cada mejora del algoritmo debe ser validada con *benchmarks* en experimentación.

II.3 Modelo de mejora de T-Coffee

En el análisis de las fases de T-Coffee determinamos, que incidir con técnicas de HPC en las fases de generación de las librería primaría y extendida, debería brindar mayor velocidad en el algoritmo, pero no influiría en la calidad de los resultados que es lo que más nos interesa por el momento cuando aumentamos a miles las secuencias de entrada. Estas dos primeras fases constan de procesos independientes (*muchos alineamientos de secuencia*) con un grado de complejidad elevado, pero que no definen el orden del alineamiento progresivo. La forma en que se construye el alineamiento se define en la fase tres (*alineamiento progresivo*) y por ende es donde se rige la calidad del resultado. Si las fases de generación de las librerías producen el mismo resultado, la calidad del alineamiento será la misma independientemente del método.

En el alineamiento progresivo el árbol guía es el encargado del orden en que se agrupan las secuencias. En la parte superior de la Figura II.5, se intenta representar con figuras y colores que

el orden en el árbol guía determina el orden de restricciones en el alineamiento; eso además se muestra en la parte inferior de la misma figura, con un ejemplo de cuatro secuencias pequeñas, que pudiesen dar alineamientos finales diferentes si fuese agrupadas de forma diferente. En T-Coffee los árboles guía comienzan a agrupar las secuencias comenzando por las más cercanas, y relacionando luego grupos anteriormente agrupados.

Cuando estamos frente a un alineamiento de miles de secuencias, el árbol es muy grande, e intentar relacionar todas las secuencias puede introducir errores. Mientras más secuencias tenga un alineamiento, habrá más tendencia a encontrar secuencias que son muy diferentes entre sí que no obstante formarán parte del alineamiento, la calidad final del resultado se verá limitada por la forma en la que estas secuencias se relacionan (*más secuencias en el mismo árbol producirá peor calidad del alineamiento global*). La prioridad que en el árbol guía se asigne a cada secuencia, dará su orden en el alineamiento final y generará el orden de *restricciones impuestas*.

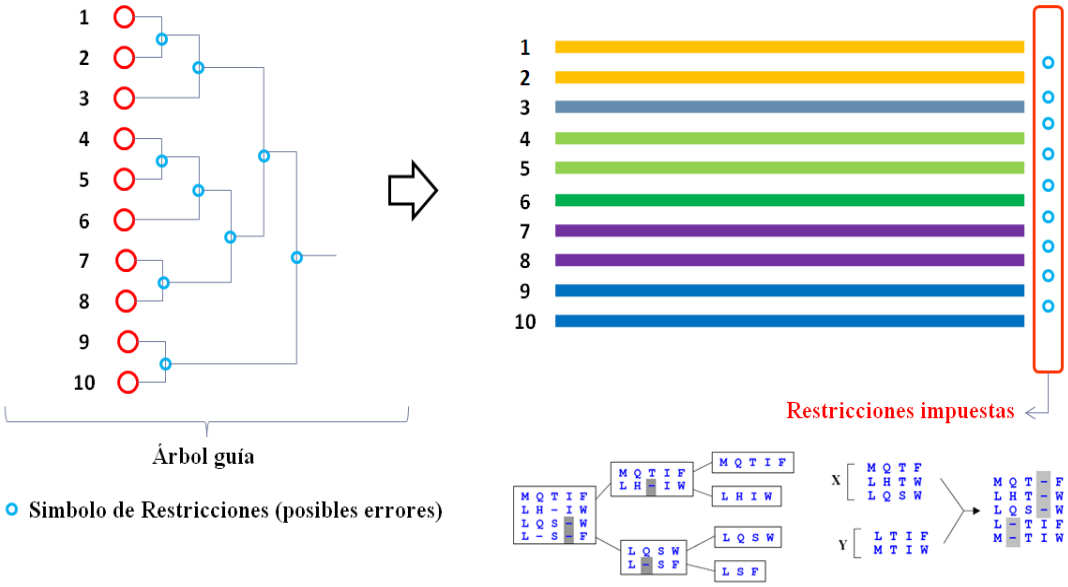


Figura II.5 Restricciones impuesta por el árbol guía en el alineamiento progresivo.

Durante el alineamiento progresivo, cada paso guiado por el árbol crea un grupo de restricciones que posteriormente no son reevaluadas. La forma en que se recorren los árboles guía en T-Coffee, es de abajo hacia arriba, o lo que es lo mismo, se comienza por las hojas. En grande árboles guía sin raíz, los vínculos entre secuencias intermedias, pueden no tener mucha relación. Este hecho, puede afectar en gran medida el alineamiento final, obteniéndose como resultado alineamientos muy largos con coincidencias aisladas entre sus componentes.

Cuando el número de secuencias se incremente al orden de miles, también sucederá que habrá grupos de secuencias que tengan más características comunes entre ellas y se diferencien de las demás. Estos grupos de secuencias se pueden ver afectados, si la relación generada por el árbol guía tiene algún error o si el árbol guía vincula secuencias distantes en pasos previos a la vinculación de secuencias más cercanas biológicamente.

Las secuencias que no estén muy cercanas evolutivamente entre sí, al llegar a la tercera fase, habrán introducido ruido en la librería extendida (*por la búsqueda de correspondencia entre pares de residuos en secuencias que no están relacionadas evolutivamente*). Sin embargo, en el presente modelo propuesto como mejora de T-Coffee en un paso previo a la librería extendida se determinarán cuáles secuencias forman grupos de secuencias cercanas evolutivamente (*están muy relacionadas entre ellas*), y se van a tratar de forma más específica, obteniéndose un alineamiento mucho más preciso. La separación de secuencias cercanas se realizará al inicio del algoritmo, empleando los métodos de análisis propios de T-Coffee.

La fase que más recursos de cómputo consume es la librería extendida, por la cantidad de relaciones (*correspondencias entre pares de residuos*) que debe analizar. Teniendo en cuenta esta característica de la librería extendida, junto a la posible formación de grupos, proponemos como modelo modificar la entrada del algoritmo, para inicialmente formar *cluster* de secuencias que muestren relación biológica realizando un análisis simple del porcentaje de identidad de secuencia.

Para cada *cluster* de secuencias se calcula el alineamiento "*intracluster*" con T-Coffee (*aunque puede ser otro algoritmo si se desea generalizar*), que posteriormente se utilizará como una nueva entrada de T-Coffee; con la diferencia de que esta última vez el alineamiento obtenido para cada *cluster* es manipulado como una secuencia única (Figura II.6), o lo que es lo mismo no se volverán a re-alinear las secuencias del *cluster*.

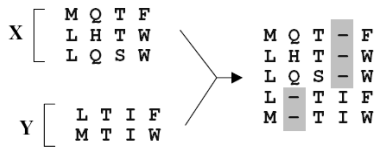


Figura II.6 Alineamientos que son analizados como secuencias únicas (*no son re-alineadas las secuencias de los alineamientos iniciales*).

Este modelo supone por una parte: la *clusterización* de las secuencias de entrada en su inicio, para resolver varios problemas con las mismas características (*ejecuciones de T-Coffee para cada cluster de secuencias*), empleando al final los resultados obtenidos en esta *clusterización* como una nueva entrada de T-Coffee (*ejecución de T-Coffee empleando como entrada los alineamientos de los clusters de secuencias*, Figura II.7); y por otra parte supone la disminución en el consumo de recurso (*tiempo de cómputo y memoria*), porque se trataría el problema de miles de secuencias, como la suma de problemas más pequeños (*por ejemplo sub-alineamientos de cien secuencias*), y como el comportamiento del tiempo de ejecución y el consumo de memoria no es lineal sino exponencial respecto al número de secuencias, siempre la suma de problemas pequeños, va a tener menor complejidad que la de un único problema de magnitud igual a la de la suma de las partes.

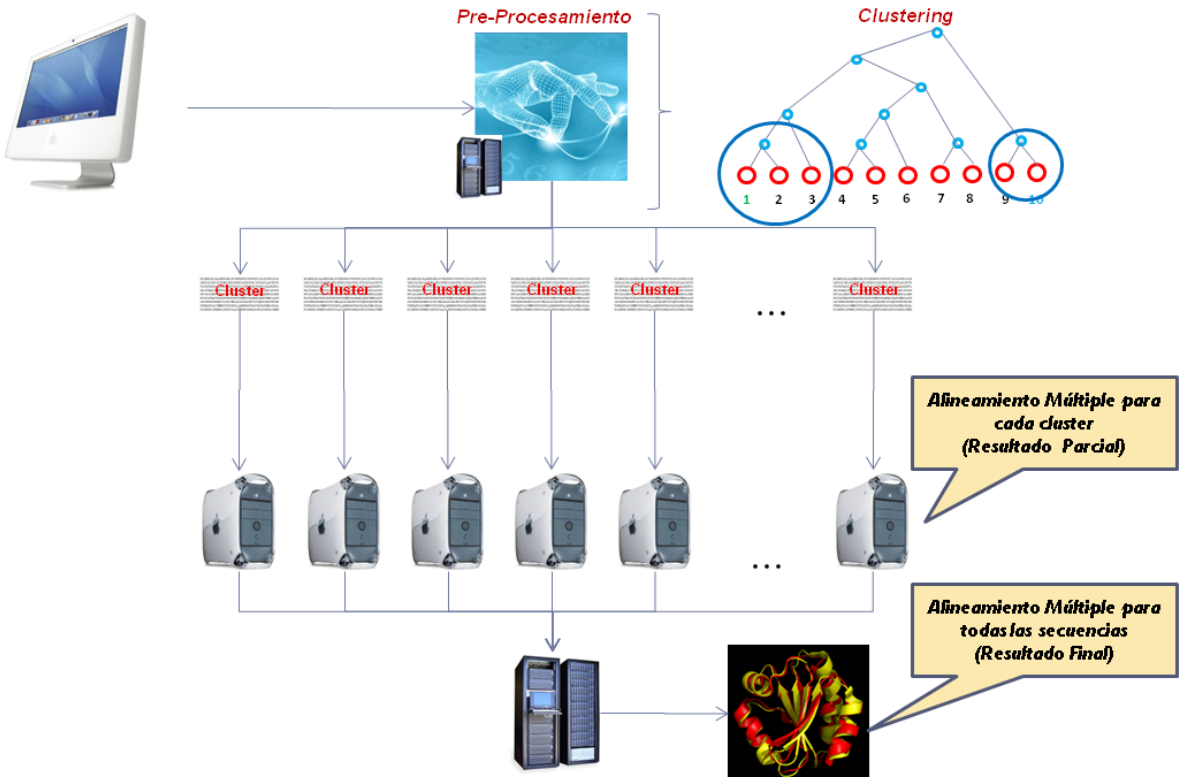


Figura II.7 Representación del posible modelo para escalar a miles de secuencias.

Desde el punto de vista biológico aumentaríamos la especificidad en el análisis y por ende el resultado debe ser mejor o igual. Desde el punto de vista de HPC, estamos proponiendo una variante en la que dividimos el problema para disminuir la complejidad en sus partes críticas y junto a esto disminuimos el consumo de recursos empleado para resolver cada una de las partes.

II.4 Modelo paralelo para T-Coffee (ClusT-Coffee)

Si comprendemos lo que supone alinear miles de secuencias una vez hayamos entendido las fases del algoritmo, nos damos cuenta que una solución para el problema de la escalabilidad es la *clusterización* (*no una simple división de los datos, sino una división con sentido biológico*). Pero para entender la *clusterización* desde el punto de vista biológico y no desde la visión matemática de segmentar un problema en partes iguales, debemos analizar lo que significa el árbol guía para miles de secuencias y como se emplea en T-Coffee para ordenar el alineamiento. Analicemos lo representado en la Figura II.8.

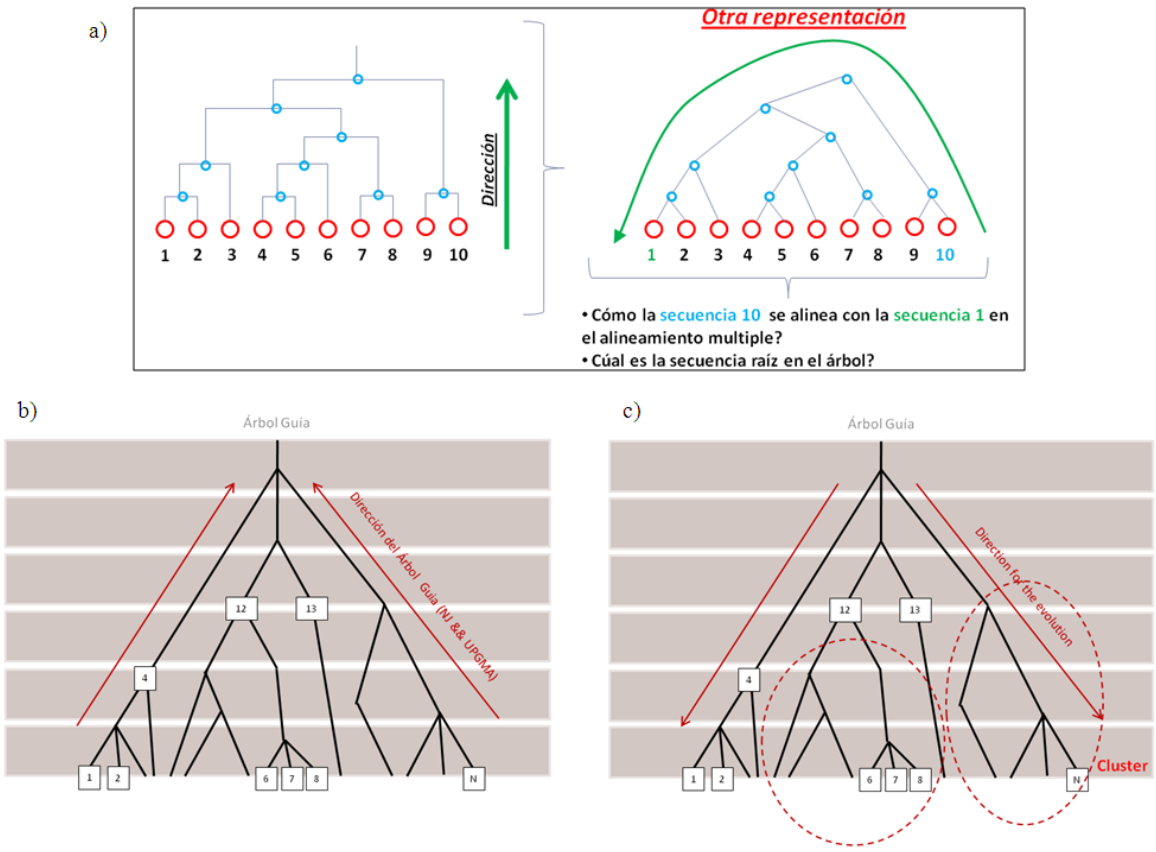


Figura II.8 Representación del análisis del árbol guía en T-Coffee.

En la Figura II.8(a), se muestra la forma en que se recorre el árbol guía en T-Coffee que es de abajo hacia arriba (*bottom-up*). Como se ha comentado anteriormente, el árbol guía es el que decide el orden en el alineamiento progresivo. De forma general los métodos empleados en T-Coffee (*método UPGMA y Neighbor-Join*), comienzan a agrupar las secuencias por las más cercanas, lo que provoca que en el alineamiento progresivo se comiencen a alinear las secuencias

más cercanas (*bottom-up*). Cada paso del alineamiento genera restricciones y si el orden en que se imponen las restricciones en el alineamientos las representásemos en el árbol (Figura II.8(b)), nos daríamos cuenta que la forma de recorrer el árbol (*bottom-up*) no es la forma natural en que la evolución impone las restricciones de cambios en las secuencias. La forma natural (*up-down*) de representación se muestra en la (Figura II.8(c)), y es lo que nos ha guiado a crear nuestra estrategia de formación de *clusters*. Cuando existan miles de secuencias a alinear, también sucede que habrá grupos de secuencias que no se vean identificados con otros (*por ejemplo en la Figura II.8(a) la secuencia 10 como representará a la secuencia 1*).

Analizando lo que representa el árbol guía en el alineamiento progresivo, decidimos formar los *clusters* de secuencias que intenten amortizar (*resolver*) la forma errónea de introducir las restricciones en el alineamiento múltiple progresivo. Para ello consideramos obtener una secuencia representativa, que fuese la que mejor se adaptase al grupo de secuencias de entrada, o la que intermediamente más se parezca y a partir de ella comenzar a formar los *cluster* de secuencias. Por el análisis de las fases en T-Coffee, en la fase de generación de la librería extendida se buscan relaciones entre todas las secuencias de entrada, por lo que decidimos ubicar al inicio la construcción de *cluster* de secuencias, para que además no introdujesen error en la generación de la librería extendida.

Realizar una división (*clusterización*) de las secuencias de entrada para hacer el análisis más preciso, requiere considerar características biológicas, junto a otros parámetros como: tamaño de los *cluster* a formar, cantidad de *cluster* y algún valor para definir que secuencias se encontrarán en cada *cluster*. Para el modelo propuesto, consideramos tomar el *porcentaje de similitud de secuencia* como valor para definir los *cluster* de secuencias, por el significado biológico que representa la similitud de secuencia. El porcentaje de similitud se obtendría a través de los mecanismo empleado por T-Coffee, y se tomaría como valor para definir que secuencias formarán parte de un *cluster*, definiéndolo como valor de corte.

Para dividir los datos según sus características existen variados métodos de *clusterización*, sin embargo decidimos crear nuestro propio método para ser específicos con el problema (*responder al significado biológico de imposición de restricciones evolutivas en el alineamiento múltiple*). Nuestra propuesta de *clusterización* de secuencias (*se basa en la observación de la forma de evolución de las secuencias en el árbol de la vida*), inicialmente parte de la construcción de una

matriz de similitud, formada por el porcentaje de similitud de los alineamientos en pares de todas las secuencias de entrada. Nos basamos en la búsqueda de una secuencia representativa, que será la que en una matriz de similitud (Figura II.9) tenga mayor promedio del porcentaje de similitud de secuencia (*para calcular el promedio se emplea el porcentaje de similitud del alineamiento de la secuencia analizada con todas las restantes secuencias*). En principio nuestro objetivo es forzar que el primer *cluster* en la *clusterización* se cree a partir de la secuencia representativa, esta secuencia dicho de alguna forma es la que mejor se acopla con las secuencias de entrada y es justificado en el porcentaje de similitud. Una vez obtenida esta secuencia representativa, se comienza el proceso de *clusterización*.

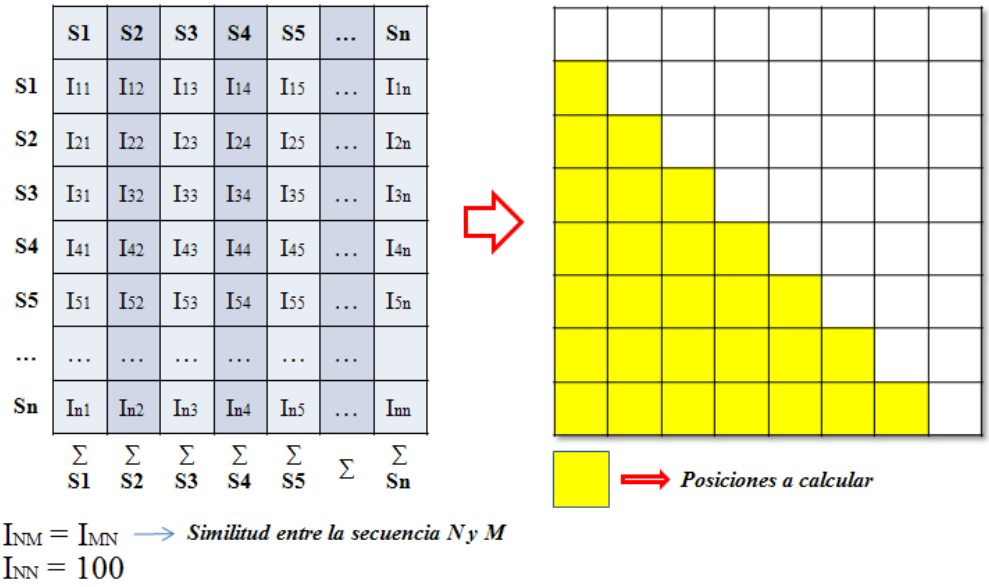


Figura II.9 Matriz de similitud de las secuencias de entrada y posiciones necesaria a calcular para construir la matriz.

El algoritmo de *clusterización* de secuencias (Figura II.10) se determina por el porcentaje de similitud en la matriz y un valor de corte determinado a priori. Todas las secuencias que se encuentren por encima de ese valor de corte o *cutoff* (*secuencias que tienen un grado de similitud superior al de corte, mientras más elevado mejor*) son sensibles a formar parte del *cluster* de la secuencia representativa.

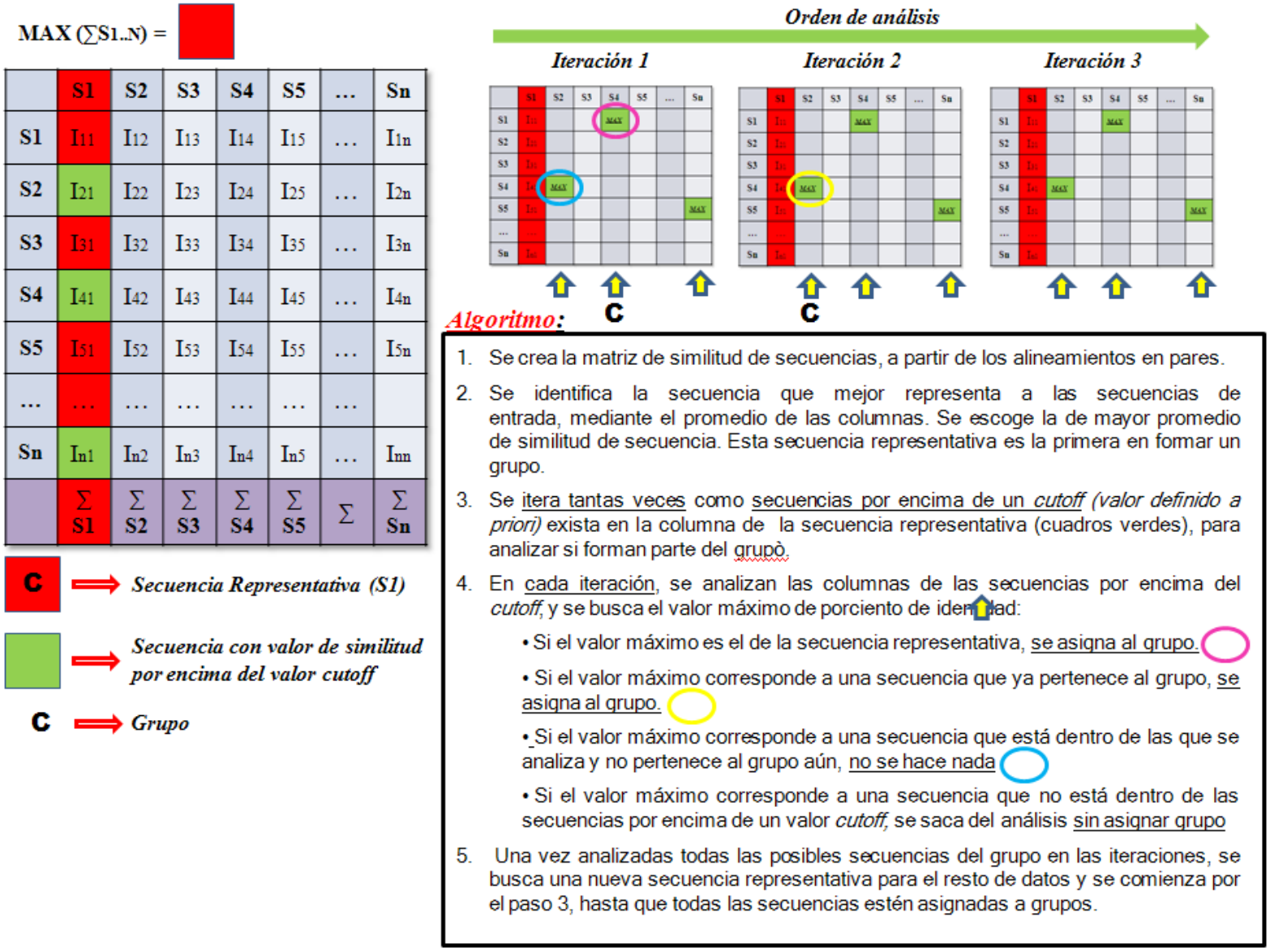


Figura II.10 Algoritmo para la formación de clusters a partir de una secuencia representativa.

Una vez definida la secuencia representativa en la *clusterización* y las sensibles a formar parte de *cluster*, se realizan tantas iteraciones como secuencias sensibles de análisis haya y en cada iteración se analiza la columna que le corresponde en la matriz de similitud a cada secuencia por encima del valor *cutoff*, con el fin de ver cuál es la secuencia que más se parece a ella (*representado en el porcentaje de similitud*) y se procede según a quien le corresponda el valor máximo de porcentaje de similitud, a asignar las secuencias sensibles al *cluster* de la secuencia representativa. Una vez analizadas todas las secuencias sensibles, se vuelve a determinar la secuencia representativa, para las secuencias restantes, y se repite el proceso hasta que todas las secuencias tengan asignado un *cluster*. Luego, se verifica que no existan *cluster* de una secuencia, en cuyo caso la secuencia sola se une con el *cluster* de la secuencia que mayor porcentaje de identidad tenga con ella. Para describir un poco mejor el proceso se presenta a continuación el *pseudocódigo* de forma general de *clusterización*:

Pseudocódigo

```
/*formación de clusters a partir de secuencias representativas para n secuencias de entrada*/

//Obtener los porcentajes de similitud para todos los pares de secuencias
para i ← 1 hasta n-1 hacer
   para j ← i hasta n hacer
      Obtener_Porcentaje_Similitud(Secuencia_i, Secuencia_j);
   fin para
fin para

cluster = 1:

//Formación de cluster de secuencias
mientras (haya secuencia sin cluster asignado) hacer

  para i ← 1 hasta n hacer
    si Secuencia_i no tiene asignado cluster entonces
      Obtener_promedio_de_porcentaje_para_secuencias_sin_cluster_asignado();
    fin si
  fin para

  Representativa =  Buscar_la_secuencia_sin_cluster_con_máximo_promedio_de_porcentaje(); //Representativa
  SecuenciasL = Secuencias_sin_cluster_con_porcentaje_similitud_con_representativa_mayor_al_umbral();

 para i ← 1 hasta longitud(SecuenciasL) hacer
    si SecuenciasL _i no tiene asignado cluster entonces
       para i ← 1 hasta longitud(SecuenciasL) hacer
          secuencia_t = Buscar_secuencia_con_porcentaje_similitud_mayor_a_la_secuencia(SecuenciasL _i);
         si secuencia_t == Representativa entonces
            A la SecuenciasL _i se le asigna el cluster de representativa;
         sino si secuencia_t se encuentra por encima del umbral y tiene asignado el cluster de la Representativa entonces
            A la SecuenciasL _i se le asigna el cluster de representativa;
         sino secuencia_t se encuentra por encima del umbral y no tiene asignado cluster entonces
            No se hace nada
         sino secuencia_t no se encuentra por encima del umbral entonces
           SecuenciasL no se continúa analizando
```

```
        fin para
    fin si
 fin para

 cluster = cluster + 1;

fin mientras

//Agrupación de los clusters con una sola secuencia
Agrupar_cluster_con_una_secuencia_con_cluster_mas_cercano();
/*fin de formación de clusters */
```

Las características de esta forma de dividir las secuencias de entrada en la fase de *clusterización* son:

1. Los tamaños y cantidad de *clusters* siempre estarán determinados por el valor de corte o *cutoff*. Este valor va a tener un significado biológico relacionado con el objetivo del alineamiento y determinará en gran medida la calidad de los resultados.
2. El algoritmo de división de secuencias lo que intenta es agrupar a las secuencias más cercanas, comenzando siempre por la que mejor represente el grupo, definiendo "*mejor*" como la secuencia que los valores de similitud promedio sean más elevados.
3. Para el mismo grupo de secuencias de entrada y el mismo valor de *cutoff*, siempre se crearán la misma cantidad de grupos, con las mismas secuencias.
4. Los *cluster*s a efectos de contraste y validación posterior, y no por imposibilidad del método propuesto no tendrán más de 100 secuencias (*posteriormente se discutirá en la experimentación*).

II.5 Paralelización de ClusT-Coffee

El modelo propuesto ClusT-Coffee desde su inicio fue diseñado de forma paralela para permitir escalar el algoritmo de alineamiento múltiple T-Coffee a miles de secuencia. En principio la forma en que hemos intentado incidir sobre el problema debe mantener la calidad de los resultados, pero eso solo podrá ser comprobado en la fase de experimentación, para ello luego plantearemos una serie de experimentos y formas para validar el modelo.

La ventaja del modelo es que al realizarse una división de las secuencias, cada *cluster* de secuencias generado, es tratado de forma independiente por el algoritmo T-Coffee y el resultado de todos los *clusters*, se emplea como una nueva entrada del algoritmo (*esta última vez tratándose como un problema más pequeño*). El diseño del modelo contempla dos fases: una

primera de pre-procesamiento de las secuencias de entrada para formar los *clusters* de secuencias, y una segunda fase de ejecución de T-Coffee para cada *cluster* de secuencias y para obtener el resultado final. La propuesta paralela se hará en base a estas dos fases del modelo. Para la implementación del modelo nos planteamos usar como modelo de programación paso de mensajes, específicamente MPI, y como paradigma de resolución el modelo *master-worker*.

II.5.1 Fase de pre-procesamiento de las secuencias de entrada

La fase de pre-procesamiento, es la parte del modelo encargada de dividir las secuencias de entrada con lo que esto conlleva (*todo el cálculo de alineamientos en pares y la formación de grupos de secuencias*), además comprende la asignación de tareas a cada uno de los nodos que se utilicen por el algoritmo. Esta fase tiene la característica de ser costosa. Requiere el alineamiento en pares de todas las secuencias de entrada, para obtener el porcentaje de identidad de secuencia de cada alineamiento. Siguiendo la misma línea antes propuesta, la ventaja es que los alineamientos en pares son independientes y pueden ejecutarse en paralelo fácilmente.

Como se comentó anteriormente el algoritmo de alineamiento global de dos secuencias tiene una complejidad de $O(N^2)$, siendo N el promedio de la longitud de las dos secuencias a alinear. Este proceso en la fase de pre-procesamiento para la *clusterización* de secuencias debe ejecutarse para las N*(N-1)/2 combinaciones de pares de secuencias, lo que aumenta la complejidad del problema y la fase de forma general. Como las N*(N-1)/2 tareas son independientes, el cómputo puede compartirse y como en T-Coffee solo son necesarias las secuencias para construir el alineamiento (*porque los demás elementos (como las matrices de puntuación) necesarios para resolver el alineamiento forman parte de la librería de la aplicación*), con pocos datos se generan grandes volúmenes de cómputo, como se demostrará en la fase experimental.

El objetivo final de ejecutar todos los alineamientos en pares es obtener un valor de similitud para cada par de secuencias alineadas, que luego formará parte de una matriz de similitud. La matriz de similitud será empleada para formar los *cluster* de secuencias más cercanas entre ellas (*descrito anteriormente*), y posteriormente se prepararán las secuencias para ejecutar T-Coffee con cada uno de los *cluster* de secuencias, que resultará en un alineamiento múltiple para cada *cluster*.

II.5.1.1 Paralelización de la fase de pre-procesamiento

Con objeto de paralelizar el modelo, e intentar utilizar lo mejor posible los nodos de cómputo, es necesario analizar el comportamiento de los datos a tratar. Como los datos biológicos con los que se trabaja no muestran regularidad, sucede que los alineamientos en pares no tomarán el mismo tiempo de cómputo, sino que dependerán de las longitudes de las secuencias a alinear. Por otra parte, una vez obtenido todos los valores para formar la matriz de similitud, los *clusters* que se formen a partir de ella, dependerán del valor de corte o *cutoff* definido a priori y de las características de los datos (*el grado de similitud de las secuencias*). Estos dos elementos son muy importantes a la hora de distribuir y asignar el cómputo a los nodos que se utilicen.

El paradigma de resolución que emplearemos como se ha comentado es el modelo ***master-worker*** y utilizaremos paso de mensajes (*específicamente MPI*), como modelo de programación. La idea es encargar al nodo ***master*** de: leer la entrada de datos (*fichero con las secuencias en formato texto*), ajustar algunos parámetros de configuración (*formato de los datos, tipo de tarea, tipo de salida de datos, valor de corte, matrices de puntuación*) y distribuir los datos de las secuencias a todos los ***workers*** junto a los parámetros de configuración definidos en el ***master***.

Anteriormente habíamos comentado, que un volumen de datos pequeño como la información de las secuencias (*es una comparación relativa*), pueden generar un grado de cómputo elevado. Teniendo en cuenta esto último junto con el comportamiento que tienen los datos y cómo afecta este comportamiento al tiempo de cómputo, decidimos como aproximación inicial; pasar a todos los nodos los datos de entrada. Estos volúmenes de datos representan pequeños costos desde el punto de vista de las comunicaciones, debido a las velocidades de interconexión de las redes actuales. Por el momento el tiempo en comunicaciones del volumen de datos que manejamos respecto al tiempo de cómputo no son comparables (*el tiempo en comunicaciones es del orden de segundos (paquetes de 1-2 MB)*, *frente a un tiempo en cómputo de minutos y horas*).

Una vez que los ***workers*** dispongan de los datos de las secuencias, tanto el ***master*** como los ***workers*** mediante un algoritmo de distribución estática decidirán sobre qué datos deben trabajar y que alineamientos en pares deben obtener. En una primera aproximación nos limitamos a distribuir la carga de cómputo según la cantidad de tareas (*considerando tarea el alineamiento de un par de secuencias*). La distribución es uniforme para cada nodo, siendo el nodo el que define el rango de alineamientos a resolver (Figura II.11). La distribución es sencilla y se calcula a

partir de una expresión dependiente de la numeración del nodo de cómputo (*definido como rank*).

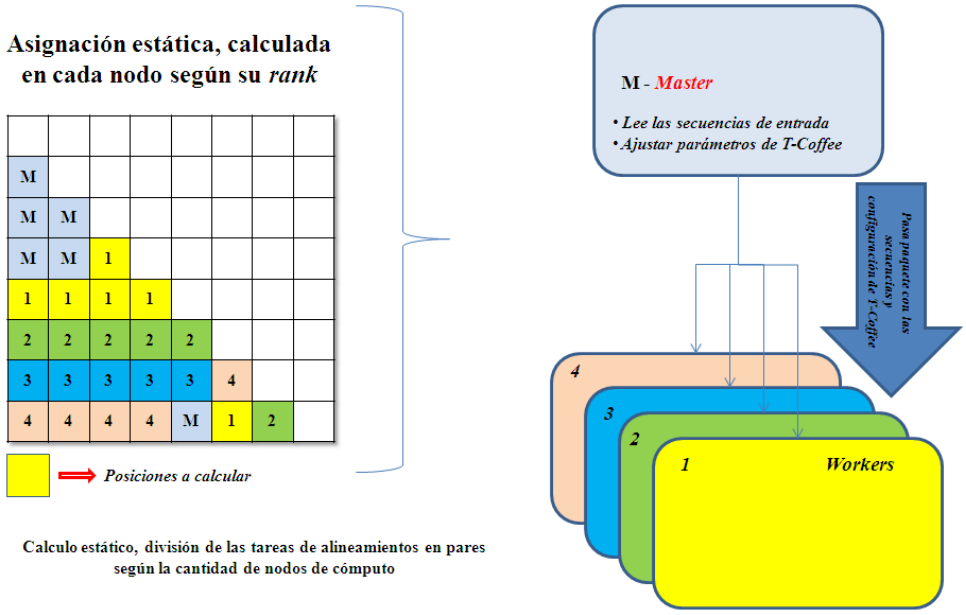


Figura II.11 Distribución estática de los alineamientos en pares en todos los nodos.

Cuando cada nodo termine de calcular los alineamientos en pares, le enviará los valores de porcentaje de identidad al ***master***. El nodo ***master*** posteriormente construye la matriz de similitud y es el encargado de obtener los *clusters* de secuencias para la ejecución del ClusT-Coffee. Cuando el nodo ***master*** haya obtenido los *clusters* de las secuencias, le pasará a cada ***worker*** un mensaje con información de los *clusters* y las secuencias que lo forman (*la información es enviada con un identificador de cada secuencia junto al cluster que le es asignado y la secuencia representativa del cluster*), los datos enviados no tendrán más información porque en cada nodo ya existe información de las secuencias.

Debido a que el tamaño y las secuencias que forman cada *cluster* dependen del valor de corte o *cutoff* definido a priori, los *clusters* no serán homogéneos en tamaño. Las técnicas para distribuir carga a nodos de cómputo por lo general se basan en una asignación uniforme de tareas y/o cómputo. En nuestro caso, no sabemos cuál será el tamaño de los *clusters*, ni las longitudes de las secuencias que lo forman, ni la cantidad de *clusters* que se formarán, lo que sí sabemos es que se utilizará un criterio de selección biológica y que el *cluster* mayor de secuencias que tendremos estará entorno a 100 secuencias (*por ser la cantidad de secuencias mayor para la que se declara*

que T-Coffee tiene resultados precisos). Las variables que en nuestro caso influyen en la carga de cómputo son, la longitud de las secuencias de los *clusters*, las cantidades de secuencias por *cluster*, la cantidad de *cluster* de secuencia y la cantidad de nodos de cómputo (***workers***). Diseñar un modelo para la distribución de estas tareas teniendo en cuenta estas cuatro variables no es una tarea sencilla. El proceso de distribución de los *clusters* de secuencias, es estático y se calcula de forma dependiente de la numeración del nodo de cómputo (*rank*), la distribución la realiza el ***master***.

La idea es sencilla y solo requiere una organización de los *clusters* según su tamaño (*definiendo tamaño como la cantidad de secuencias de cada grupo*), y la asignación a cada nodo de cómputo de forma ordenada comenzando por las mayores (Figura II.12).

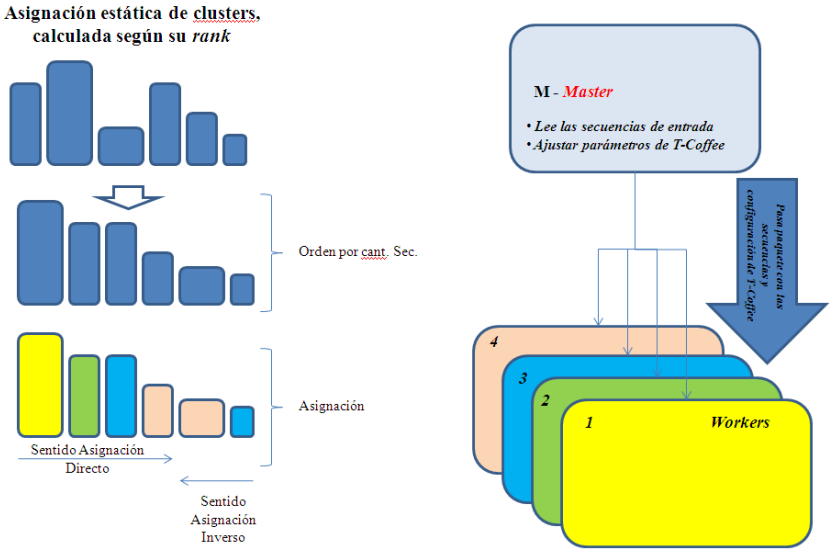


Figura II.12 Distribución estática de los grupos de secuencias a los nodos ***workers***.

El proceso de asignación de *clusters* se calcula en el nodo ***master*** y luego se le pasa a los ***workers*** esta información. El algoritmo inicialmente ordena todos los *clusters* de secuencias según la cantidad de secuencias. Luego se comienza a asignar de forma ordenada a cada ***worker*** un *cluster* comenzando por el mayor, hasta que todos tengan asignado un *cluster*. Una vez todos los ***workers*** tenga asignado un *cluster*, se comienzan a asignar en orden inverso de los ***workers***. Al último ***worker*** le es asignado el *cluster* de secuencias que quedó pendiente y que en su caso será el mayor de los que queden, luego se siguen asignando *clusters* a los ***workers*** en orden inverso, siempre sumando el nuevo *cluster* al nodo que minimice la diferencia por el análisis de

sus nodos vecinos. El pseudocódigo de forma reducida de la distribución de los *cluster* se muestra a continuación:

Pseudocódigo

```
/*asignación de n clusters a los size nodos de cómputo*/

//Ordenar los cluster por tamaños
Ordenar_los_cluster_por_tamaño_comenzando_por_el_mayor();

 count_size =0;
 para i ← 1 hasta n hacer
   count_size = count_size + 1;
   si i < count_size entonces
       Asignar_el_cluster_i_a_el_nodo_count_size();
   sino
      Buscar_el_nodo_con_menos_carga_en_orden_inverso_y_asignarle_el_cluster_i();
   fin si
fin para
```

Un factor que limita este paso es, que existan menos *cluster* que nodos de cómputo, en cuyo caso los nodos contribuirán en el cálculo de la matriz, pero no en la obtención de alineamientos. En el proceso de asignación de *clusters* a los nodos de cómputo (*calculado en el master*), cuando el ***master*** haya terminado, le pasará la información de asignación a los ***workers***. Teniendo la información de los *cluster* de secuencias y la asignación de *cluster* a los nodos, los ***workers*** podrán calcular a cada *cluster* el alineamiento múltiple según corresponda. En este momento comienza la segunda fase de ejecución del algoritmo propio de T-Coffee.

II.5.2 Fase de ejecución de T-Coffee para cada *cluster* de datos

En el momento que se han definido los *cluster* de secuencias cercanas y que se hayan asignado a los nodos de cómputo, comienza a ejecutarse de forma iterativa el algoritmo T-Coffee en cada uno de los nodos ***workers*** para los *cluster* de secuencias asignados. Cada iteración en un ***worker*** significa la ejecución de T-Coffee para un *cluster* de secuencias. Al finalizar cada iteración los ***workers*** le envían los resultados al nodo ***master***, que va almacenando los alineamientos múltiples de los *cluster* de secuencias, para posteriormente emplearlos a todos como una última entrada de T-Coffee que generará el alineamiento múltiple final.

El nodo ***master*** en esta fase, hasta que no recibe el alineamiento múltiple de todos los ***workers*** lo hace es esperar los mensajes con los resultados de los alineamientos. Una vez obtenidos todos los

alineamientos, el nodo ***master*** ejecuta nuevamente T-Coffee utilizando cada alineamiento como si fuese una secuencia única (*comentado con anterioridad*) para generar el alineamiento final.

Esta fase es dependiente del cálculo de los alineamientos por cada uno de los nodos. En un nodo cada iteración como máximo consumirá en recursos lo equivalente a un problema de un centenar de secuencias con las mismas características que las secuencias de entrada. Por la forma en que se ha comprendido el problema (*clusterización de las secuencias que más se asemejan*) cada uno de los *cluster* de secuencias debe obtener un alineamiento más específico que luego se mantendrá en el alineamiento final. Lo que se pretende en el modelo es mantener la calidad de los resultados, y ese es el motivo por el que no se crearon *cluster* de secuencia con tamaño uniforme que contribuyesen a un mejor balance de la carga de cómputo en los nodos.

Una representación secuencias del modelo se muestra en la Figura II.13. Con el modelo, aumentamos a miles las secuencias a procesar por el algoritmo, intentando reducir con un significado biológico el orden de entrada de las secuencias, manteniendo además la calidad de los resultados del algoritmo. El grado de reducción de las secuencias de entrada en el modelo será dependiente del valor de corte o *cutoff* que se emplee para obtener los *cluster* de secuencias y que dependerá del grado de especificidad biológica con el que se desee procesar cada *cluster* de secuencias entradas.

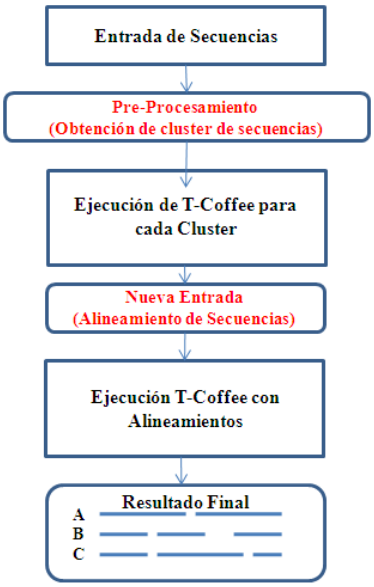


Figura II.13 Modelo propuesto para aumentar la cantidad de secuencias de entrada de T-Coffee a miles y disminuir el tiempo de cómputo y consumo de recursos.

II.6 Propuesta de validación de ClusT-Coffee

El objetivo del modelo propuesto es poder escalar los algoritmos de alineamiento múltiple de secuencias a miles, manteniendo la calidad de los resultados superando limitaciones actuales. En nuestro caso por haber tomado a T-Coffee como propuesta de estudio, la validación del modelo se realizará contra T-Coffee. Por lo tanto, en los experimentos que diseñemos, mantendremos los mismos parámetros en nuestro modelo con T-Coffee, que en T-Coffee original y compararemos los resultados de acuerdo a la función de puntuación utilizada por él. Como además en ocasiones las funciones empleadas para evaluar la calidad de los alineamientos no reflejan cuán correcto pueden estar, emplearemos el benchmark Balibase (Anexo I), que contiene el resultado correcto dado una entrada de secuencias, con la que se puede comparar la calidad del alineamiento obtenido.

Desde el punto de vista de HPC, los experimentos diseñados se centrará en el consumo de recursos (*tiempo de cómputo y memoria*), partiendo de que este es uno de los problemas graves presentados por el algoritmo actual. La experimentación para HPC se realizará con secuencias sintéticas (*construidas de forma aleatoria por un programa*).

CAPITULO III

Experimentos para la validación de ClusT-Coffee

En el presente capítulo se presentarán y discutirán una serie de experimentos para poder validar el modelo paralelo propuesto para T-Coffee. Inicialmente se estudiará el comportamiento de T-Coffee a través de un grupo de experimentos de rendimiento, posteriormente se mostrarán los resultados obtenidos por el modelo desde la visión biológica para verificar la calidad de los resultados, y finalmente se presentan resultados de rendimiento para el modelo propuesto.

III.1 Introducción

Validar el modelo propuesto para escalar el algoritmos de alineamiento múltiple de secuencias T-Coffee, requiere por una parte un análisis biológico y por otra computacional. Desde el punto de vista biológico lo más importante es verificar si podemos mantener la calidad de los resultados, a medida que crece el conjunto de secuencias de entrada, que es una de las problemáticas más grandes de este tipo de algoritmos. En base a este aspecto decidimos utilizar la base de datos Balibase 3.0 (*benchmark*) como patrón de comparación. Desde el punto de vista de HPC, se enfocó a disminuir el tiempo de cómputo y el consumo de memoria, que es otro de los problemas del algoritmo y fue en base a lo que se preparó otro grupo de experimentos.

El entorno de ejecución utilizado para realizar los experimentos fue un cluster de 8 nodos con sistema operativo Linux CentOS Release 4.7. Las características del hardware de cada nodo se muestran en la Tabla III.1.

Tabla III.1 Características del hardware de los 8 nodos de cómputo del cluster.

Componentes	Valor	Observaciones
Procesador	Pentium-D	3.4 Hz
CPU cores	2	-
Cache L2	2 x 2 MB	Non-shared
RAM	2 x 512 MB	Dual-chanel
HDD	80 GB	SATA

Las mediciones de tiempo se realizaron con funciones de la librería estándar de C, incluidas en el código de T-Coffee y las mediciones de memoria se realizaron con la aplicación *top* del sistema operativo. Se realizaron varios experimentos y los valores mostrados en todos los casos son valores medios obtenidos.

La metodología seguida en la elaboración de los experimentos fue:

1. Ejecución del algoritmo T-Coffee original para analizar el consumo de recursos de cómputo: Se pretendió estudiar la complejidad del algoritmo a partir de experimentos en los que se variaba la cantidad de secuencias de entrada. Para realizar este experimento se tomaron secuencias sintéticas, para definir una longitud exacta de las secuencias, debido a que la complejidad del alineamiento depende de la longitud de la secuencia. En estos experimentos se midió el tiempo de cómputo y el consumo de memoria en las dos primeras fases del algoritmo T-Coffee (*construcción de la librería primaria y extendida*), que son las que más memoria consumen.
2. Ejecuciones del modelo propuesto para paralelizar T-Coffee y ejecuciones de T-Coffee original, todas las ejecuciones con la misma entrada a los dos algoritmos, con el fin de analizar:
    a. **Calidad de resultados** – Se tomaron entradas combinadas de secuencias de varias familias de la base de datos Balibase (*se explicará más adelante cuando se presente el experimento*) para diferentes comportamientos de secuencias biológicas (*diferentes valores de similitud entre las secuencias de una misma familia, se emplearon secuencias de proteínas*) y se compararon los resultados biológicos obtenidos en ambos casos (*el modelo propuesto y T-Coffee original*). Se utilizó como métrica la misma función que emplea el algoritmo T-Coffee para determinar la calidad del alineamiento múltiple.

b. **Cómputo** – Para una entrada de mil secuencias, se realizaron ejecuciones con diferentes cantidades de nodos de cómputo, además se ejecutó T-Coffee original con la misma entrada. Se realizaron mediciones de tiempo de cómputo y de consumo de memoria. Se tuvieron en cuenta las fases presentadas en el modelo.

Los experimentos presentados en este capítulo se describirán, se justificará el objetivo para cada uno, se presentarán los resultados y luego se discutirán. Al final se presentará un análisis general del proceso de experimentación.

III.2 Mediciones de tiempo y consumo de memoria en T-Coffee

Para validar el modelo propuesto para T-Coffee, primero realizamos un grupo de mediciones de rendimiento a la aplicación original. El objetivo fue analizar la complejidad del algoritmo descrita teóricamente y definir los límites en el consumo de memoria del algoritmo, para conocer cuánto puede afectarnos posteriormente en la creación de los grupos de secuencias empleados por el modelo. Como T-Coffee define que su ejecución comienza a no ser muy buena para más de 100 secuencias, que además fue la razón por la que definimos el valor máximo de secuencias para la formación de los grupos en el modelo propuesto, consideramos hacer experimentos para grupos de entrada que variasen en 10 secuencias comenzando por 10.

Los experimentos se pensaron hacer hasta llegar a valores en los que la aplicación consumiese toda la memoria del sistema y tuviese que emplear la memoria *swap*. La longitud de las secuencias se decidió que fuese 500, y por este motivo se generaron de forma sintética, para garantizar que todas tuviesen la misma longitud. La longitud se mantuvo constante para poder aumentar uniformemente el tamaño el problema, puesto que la complejidad depende de la longitud de las secuencias que se pretenden alinear.

Las mediciones se tomaron para cada una de las fases principales del algoritmo, y se tuvo en cuenta el tiempo de cómputo y porcentaje de utilización de memoria. Las secuencias sintéticas se obtuvieron a partir de un generador de secuencias automáticas nombrado ***rMotifGen*** (http://bioinformatics.louisville.edu/brg/rMotifGen/index.htm)

Se realizaron 20 experimentos para cada una de las entradas con tamaños de cantidad de secuencias diferentes. Se obtuvieron mediciones de tiempo de cómputo para *la librería primaria*, *la librería extendida*, *el alineamiento progresivo* y el *tiempo total*, además se midió el consumo

de memoria máximo alcanzado en la librería primaria y en la librería extendida. Se obtuvo la media de los resultados para descartar posibles errores en el sistema y se graficaron. Los resultados se muestran en la Tabla III.2 y la Figura III.1

Tabla III.2 Tabla con los resultados obtenidos en las ejecuciones del algoritmo de alineamiento múltiple de secuencias T-Coffee utilizando como entrada secuencias sintéticas de 500 residuos.

Cantidad de Secuencias	Tiempos (Segundos)				% Memoria Usada (MB)	
	Lib. Primaria	Lib. Extendida	Alineamiento	Total	Lib. Primaria	Lib, Extendida
10	8	18	1	27	0,6	3,4
20	32	77	5	114	0,7	4,4
30	77	180	12	269	0,9	7,1
40	139	333	22	494	1,2	12,3
50	230	521	41	792	1,5	17,5
60	343	764	67	1174	1,9	22,1
70	482	1059	98	1639	2,3	31,5
80	914	1992	190	3096	2,9	40,3
90	1242	2563	271	4076	3,5	50,2
100	1370	2760	372	4502	4,2	60,2
110	1468	2832	384	4684	5	73,5
120	1843	3437	490	5770	5,8	SWAP
130	3182	6061	907	10150	6,6	SWAP

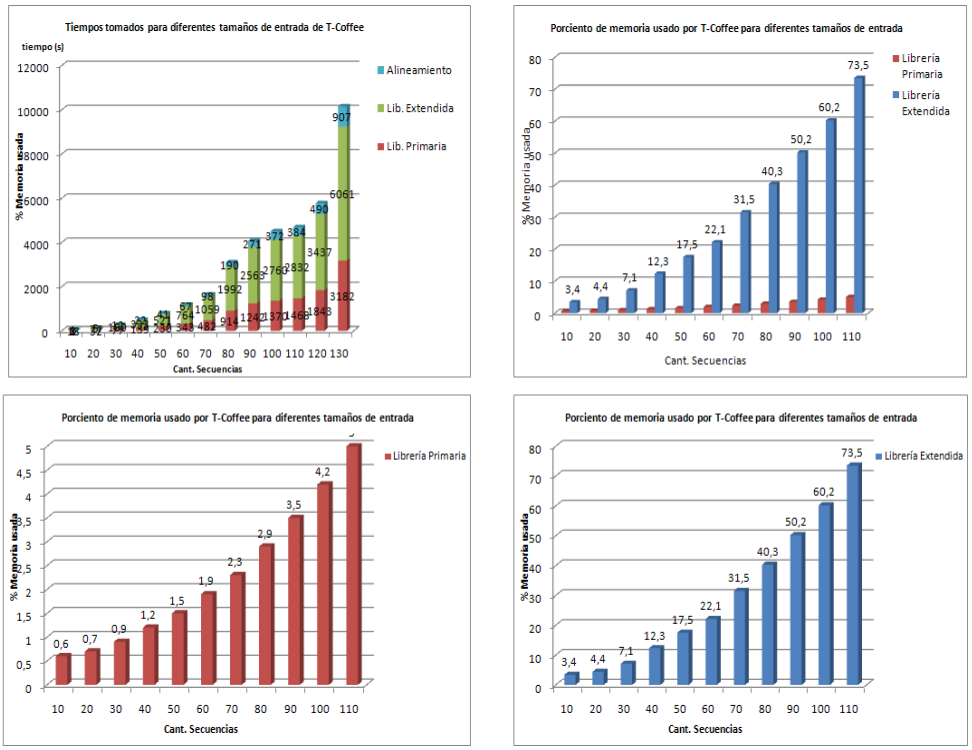


Figura III.1 Gráficos de medición de tiempo y porcentaje de consumo de memoria para las ejecuciones de T-Coffee utilizando como entrada secuencias sintéticas de 500 residuos.

Los resultados obtenidos en las ejecuciones de T-Coffee nos mostraron que:

- ✓ El consumo de memoria crece exponencialmente, como se describe en cada una de las fases del algoritmo.
- ✓ Hay una diferencia notable entre los porcentajes de consumo de memoria de la librería primaria y la librería extendida, siendo el cómputo de la librería extendida el responsable de que se fuese a la swap para 120 secuencias de 500 residuos.
- ✓ El tiempo de ejecución como describe el algoritmo tiene un comportamiento exponencial. Y las etapas que más consumen tiempo son la de creación de la librería.
- ✓ La gráfica nuestra un cambio brusco y muy notable en tiempo cuando se llega a 130 secuencias de 500 residuos, que es donde se consumió toda la memoria y se fue a la *swap*.

De estos resultados podemos concluir, que:

- ✓ La progresión de tiempo de cómputo y consumo de memoria respecto al número de secuencias a analizar es exponencial, y que para valores no muy mayores a las 100 secuencias el algoritmo supera 1 GB de consumo de memoria en la fase de la librería extendida. Este resultado nos índica que el comportamiento óptimo para T-Coffee son 100 secuencias.
- ✓ La granularidad máxima de *cluster* de secuencias es 100 para seguir usando T-Coffee original como método.

Apreciamos de forma general, antes de continuar con los experimentos que T-Coffee necesita recursos y está limitado por los recursos disponibles a medida que crecen las secuencias de entrada. Verificamos que T-Coffee realmente consume muchos recursos de cómputo que lo vuelve inoperable para el orden de miles de secuencias, por lo que hay margen para optimizar, con la utilización de *clusters* de secuencias.

III.3 Experimentos para medir la calidad de los resultados de ClusT-Coffee contra T-Coffee.

Una vez estudiado el comportamiento del algoritmo T-Coffee a través de los resultados obtenidos en los experimentos, vamos a analizar la calidad de los resultados de los alineamientos obtenidos por el modelo. Hasta el momento uno de los mayores problemas de los algoritmos de

esta categoría es que cuando aumenta el número de secuencias disminuye la calidad de los resultados. Aunque es complicado comprobar la calidad de un algoritmo para miles de secuencias, porque el resultado no se puede interpretar tan fácilmente (*dado un número de secuencias a alinear no se conoce la mejor solución, no existe una única solución correcta*). En nuestro caso como el objetivo principal es aumentar la escala de entrada de datos del algoritmo manteniendo la calidad actual de los resultados, conformamos estos experimentos de medición de la calidad, en un paso previo a la implementación del modelo paralelo, visto desde una versión secuencial donde ya se ha separado la generación de *clusters* como fase en la ejecución (*realizando las operaciones en iteraciones*).

Para poder validar el algoritmo nos planteamos dos puntos:

- ✓ El alineamiento para grupos pequeños de secuencias de entrada no podía ser peor que el algoritmo original.
- ✓ Se obtuvieron varias entradas de validación (*Balibase*) cuya calidad se conocían a priori y se analizó el comportamiento del algoritmo para definir los grupos y obtener el alineamiento final. Utilizando esta entrada el algoritmo debía mejorar o igualar en el peor de los casos al original.

Teniendo en cuenta los aspectos anteriores, a partir del *benchmark* Balibase 3.0 conformamos un grupo de ficheros para medir la calidad de nuestro modelo frente a T-Coffee original. El *benchmark* Balibase 3.0, está conformado por fichero de secuencias de familias de proteínas con diferentes características en su comportamiento (*porcentaje de similitud diferente en las familias*). Para cada una los comportamientos representados en el *benchmark* se consideró un ejemplo, que fue ejecutado por T-Coffee original y por el modelo. Cada uno de los ejemplos conformados para un comportamiento se diseño mezclando diferentes secuencias de familias, para intentar ver si el modelo con el agrupamiento lograba mejorar los resultados y relacionar las secuencias con sus familias.

Como la *clusterización* de secuencias en el modelo es dependiente del valor de corte que se decida a priori, y como esto en el algoritmo será configurado por el usuario, definimos varios valores de corte para intentar ver la influencia del parámetro en el procesamiento de las secuencias. El conjunto de secuencias que se tomó para verificar los resultados es de 50, porque

a pesar de que se va a escalar el algoritmo a miles, en esta fase queremos validar el comportamiento respecto a T-Coffee original no saturado.

Para entender el comportamiento de los gráficos debemos definir los siguientes términos:

cutoff: El valor *cutoff* presentado en las gráficas es el porcentaje de similitud de secuencias empleado para la formación de los *clusters*.

score: Valor de puntuación empleado por T-Coffee para determinar la calidad del alineamiento múltiple (*tiene un significado biológico y es mejor mientras mayor sea*).

Las clasificaciones de los grupos de proteínas empleados se muestran en la Figura III.2, siendo cinco tipos de referencias que evalúan distintas configuraciones de familias de proteínas.

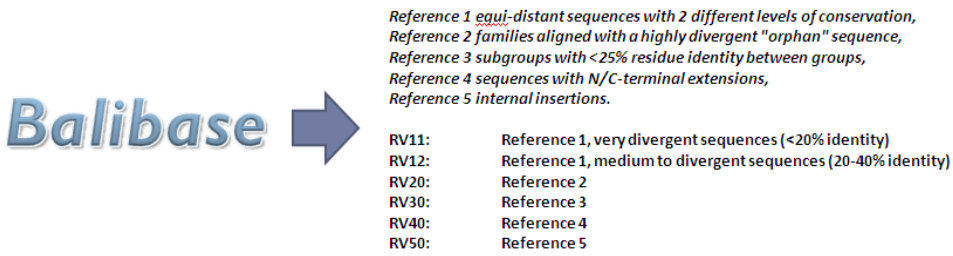


Figura III.2 Clasificación de los grupos de familias de proteínas contempladas en Balibase.

Los resultados obtenidos para cada uno de los casos se muestran en la Figura III.3 a la Figura III.8.

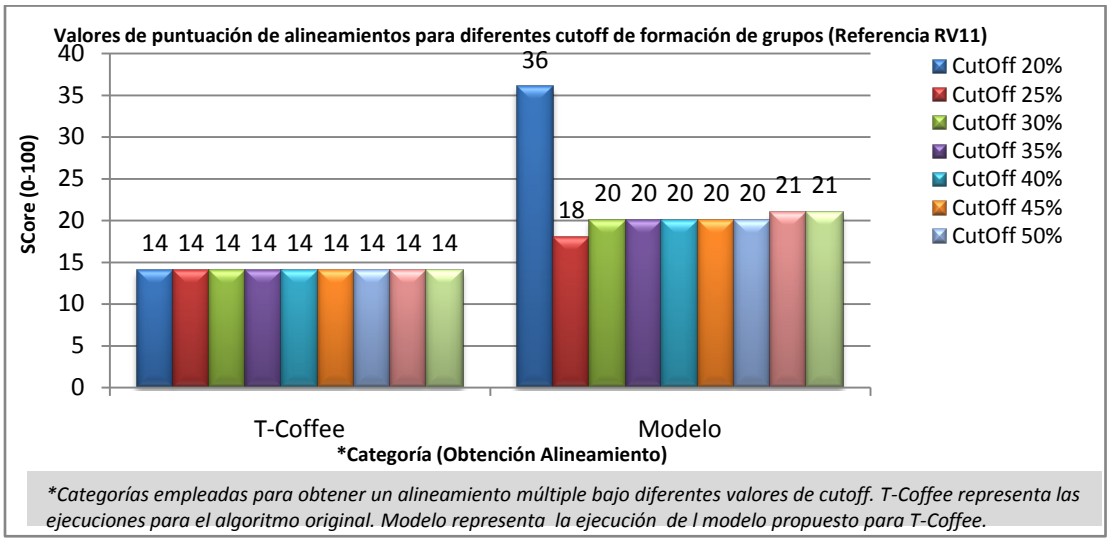
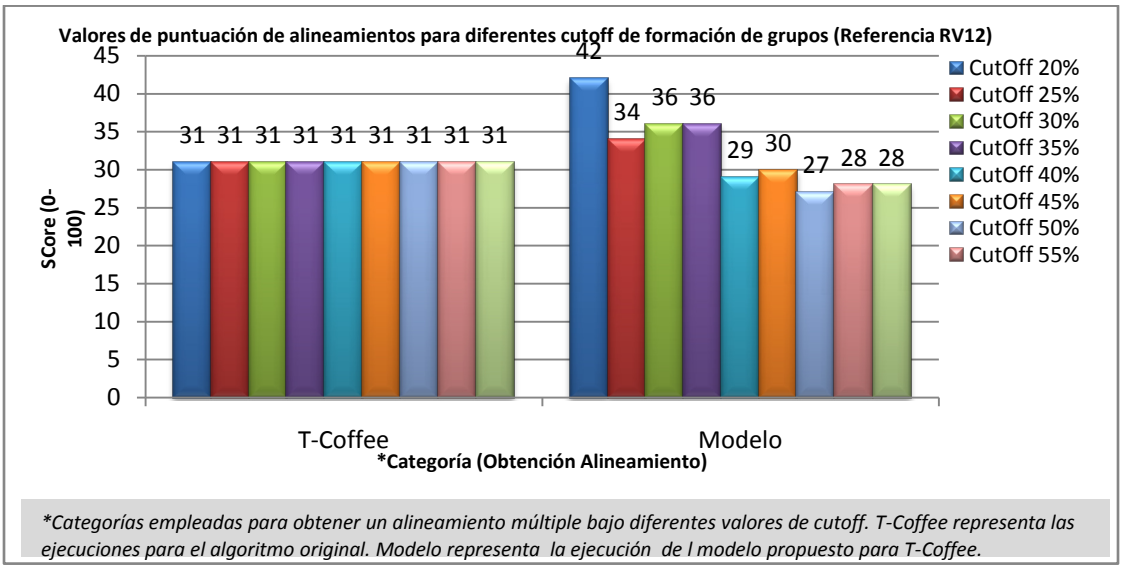


Figura III.3 Gráficos de medición de calidad de los resultados para las ejecuciones de T-Coffee y el modelo propuesto utilizando como entrada secuencias de la referencia RV11.

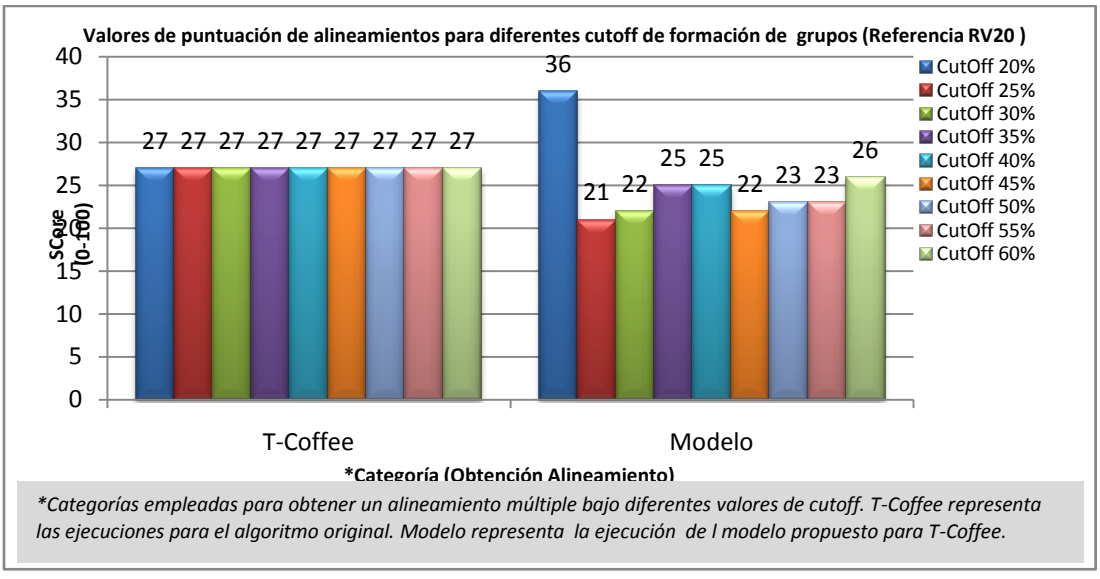
El gráfico anterior (Figura III.3) muestra un incremento en la calidad del alineamiento obtenido para todos los valores de *cutoff* empleado. Se destaca el resultado obtenido para clusters formados con un valor de *cutoff* de 20% de porcentaje de similitud de secuencia.



**Categorías empleadas para obtener un alineamiento múltiple bajo diferentes valores de cutoff. T-Coffee representa las ejecuciones para el algoritmo original. Modelo representa la ejecución de l modelo propuesto para T-Coffee.*

Figura III.4 Gráficos de medición de calidad de los resultados para las ejecuciones de T-Coffee y el modelo propuesto utilizando como entrada secuencias de la referencia RV12.

El grupo de secuencias del experimento anterior (Figura III.4) fue obtenido por la unión de secuencias de familias de proteínas con porcentaje de similitud entre el 20% y 40%. Se observa en el gráfico una mejora en los resultados para valores de *cuttof* entre 20% y 35%.



**Categorías empleadas para obtener un alineamiento múltiple bajo diferentes valores de cutoff. T-Coffee representa las ejecuciones para el algoritmo original. Modelo representa la ejecución de l modelo propuesto para T-Coffee.*

Figura III.5 Gráficos de medición de calidad de los resultados para las ejecuciones de T-Coffee y el modelo propuesto utilizando como entrada secuencias de la referencia RV20.

El gráfico anterior muestra una mejora para el grupo de proteínas formado de la categoría RV20, para valores de porcentaje de similitud del 20%. Para los demás valores de *cutoff* no se presenció mejora.

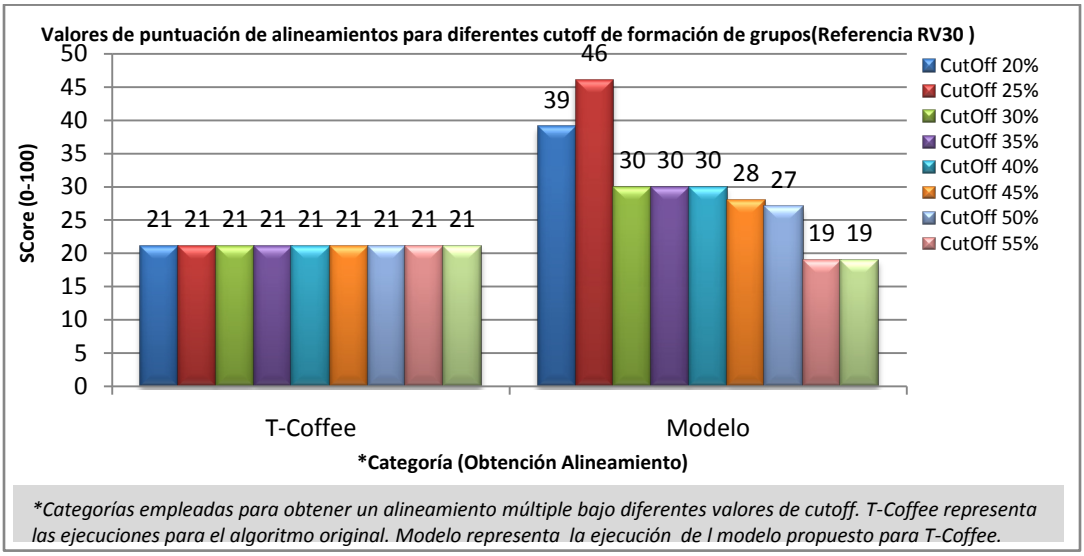


Figura III.6 Gráficos de medición de calidad de los resultados para las ejecuciones de T-Coffee y el modelo propuesto utilizando como entrada secuencias de la referencia RV30.

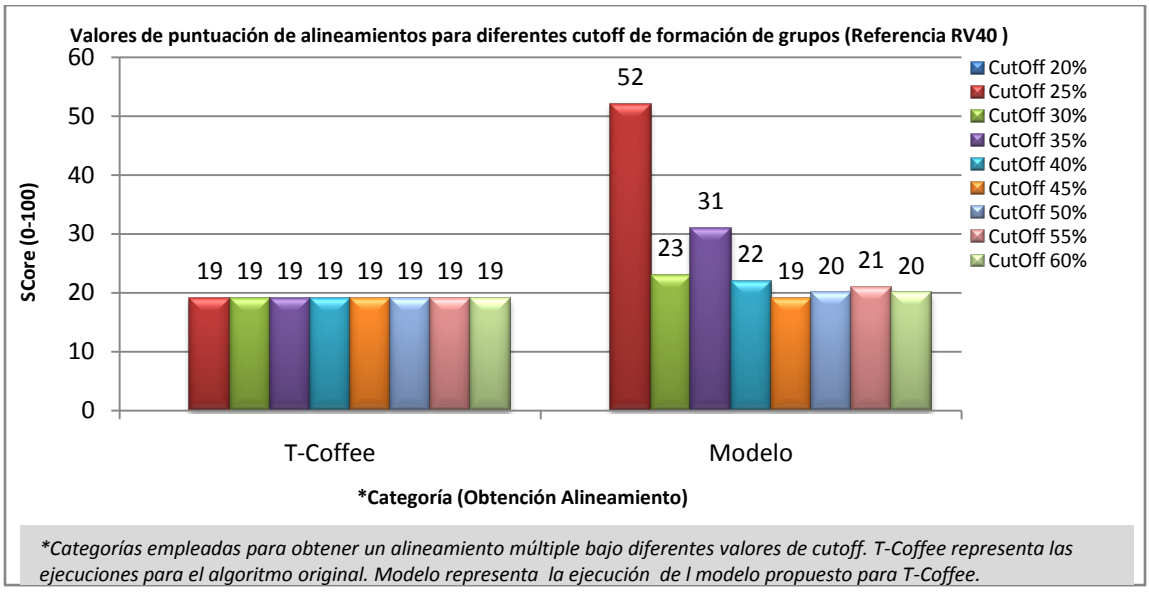


Figura III.7 Gráficos de medición de calidad de los resultados para las ejecuciones de T-Coffee y el modelo propuesto utilizando como entrada secuencias de la referencia RV40.

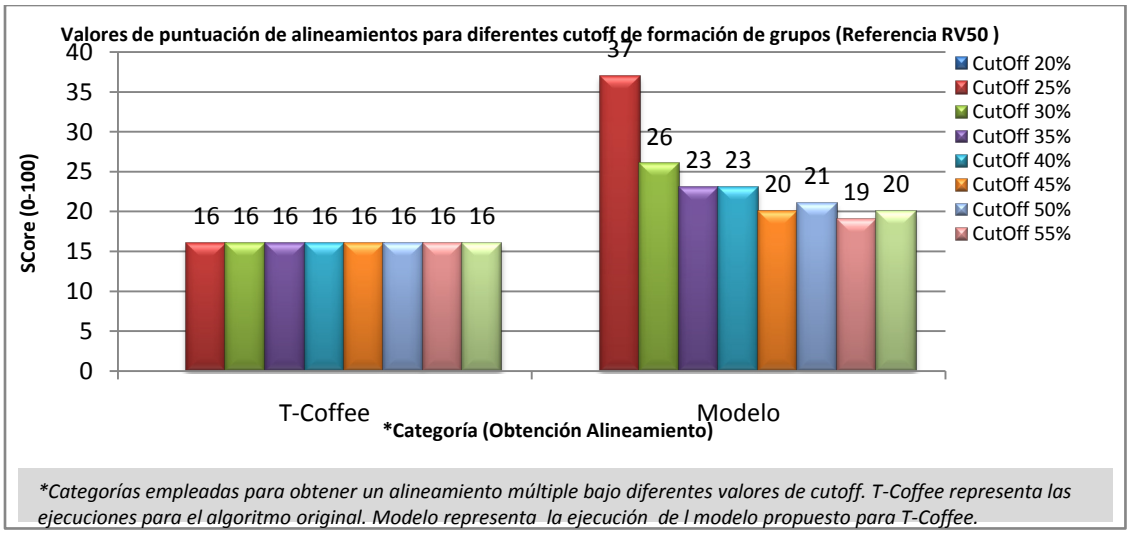


Figura III.8 Gráficos de medición de calidad de los resultados para las ejecuciones de T-Coffee y el modelo propuesto utilizando como entrada secuencias de la referencia RV50.

De forma general para los grupos de proteínas formados de las referencias de la base de datos Balibase, se obtuvieron por la clusterización en el modelo mejora respecto al T-Coffee original, aunque no para valores de *cutoff* constantes.

Los resultados obtenidos y representados en las gráficas anteriores muestran que:

- ✓ Para cada una de las categorías estudiadas, la formación de los *cluster* de secuencias a la entrada del modelo mejoran los valores de puntuación *score*.
- ✓ Que los resultados obtenidos para un mismo fichero de entrada pueden variar significativamente por variaciones del valor de corte utilizado para agrupar las secuencias.
- ✓ En los dos últimos casos (*no se observan resultados para valores de corte 20%*), se observa que tener valores de corte muy bajos pueden provocar que no exista agrupamiento de las secuencias por especificidad. Esto lo provoca que todas las secuencias tienen porcentajes de similitud superior al 20%, por lo que todas caen dentro del mismo cluster de secuencias.
- ✓ Se logra en algunos casos aumentar en gran medida la calidad del alineamiento.

Estos resultados corroboran lo planteado cuando se diseño el modelo y la forma en que se obtendrían los grupos de secuencias, que es que lo datos pueden tener características diferentes y en dependencia de sus propiedades, los grupos que se formen bajo un valor de *cutoff* tendrán un resultado u otro. Este comportamiento muestra que es aconsejable definir un modelo, que en

dependencia de la similitud en los datos de entrada y recogidos en la fase inicial en la matriz de similitud, se puede decidir qué valor de corte es mejor para intentar maximizar la calidad del alineamiento a partir de la adaptación a las características de los datos de entrada.

Las gráficas mostraron que el modelo se *clustering* de los datos es factible, mejorando de alguna forma la calidad del algoritmo original que es el mejor hasta el momento en su categoría. Y dicho esto, es comprensible gastar esfuerzos en intentar con técnicas de paralelismo disminuir el tiempo de cómputo y el consumo de memoria, porque si el comportamiento del modelo a baja escala mejora el original, cuando estemos frente a miles de secuencias, el agrupamiento de secuencias mediante el modelo presentado permite aumentar el número de secuencias sin penalizar el tiempo de ejecución y el uso de memoria.

III.4 Experimentos para comparar rendimiento ClusT-Coffee contra T-Coffee.

Después de haber obtenido los resultados de los experimentos anteriores, se decidió hacer mediciones del rendimiento del modelo frente a T-Coffee, para resolver un alineamiento de 1000 secuencias. Para realizar este experimento se creó un fichero con 1000 secuencias reales de longitud variable, las características de las secuencias se muestran en la Tabla III.3.

Tabla III.3 Características de las secuencias empleadas en el experimento

Longitud mínima de la secuencia	*Longitud máxima de la secuencia*	*Promedio de longitud de las secuencias*
19	254	159

Por los resultados obtenidos en el primer experimento esperábamos que la ejecución en T-Coffee original para las 1000 secuencias, tomase algunos días, pero sucedió que la ejecución al cabo de 7 días (*ocurrió en varias ocasiones*) se detenía, porque el proceso era terminado de forma abrupta por el sistema, por consumir toda la memoria disponible. En otras palabras T-Coffee para resolver este problema consumió en el entorno de ejecución 1 GB de memoria RAM y toda la memoria *swap* que le permitió el sistema operativo hasta el momento que fue terminado. Este hecho no nos permite realizar una comparación directa con T-Coffee pero si tomaremos como referencia los días que demoró en la ejecución T-Coffee antes de ser terminado.

Los experimentos desde el punto de vista del modelo se presentan considerando el tiempo de ejecución de las dos fases del modelo (*pre-procesamiento y obtención de los alineamientos*)

propuesto y además se consideró el tiempo de ejecución total. Se presenta un ejemplo para describir el comportamiento del algoritmo en la obtención de grupos empleando 8 nodos de cómputo. Y se analiza la escalabilidad de la fase 1 que es de la que más tiempo de cómputo consume.

III.4.1 Tiempo de ejecución

Primero presentemos los resultados obtenidos en las mediciones de tiempo de cómputo para diferentes cantidades de nodos de cómputo (Tabla III.4). Los resultados presentados consideran el tiempo total en segundos, el tiempo que se tarda en pre-procesar la entrada para obtener los *clusters* de secuencias, y lo que se tarda en obtener los alineamientos de los grupos y el alineamiento final. De forma gráfica se muestran en la Figura III.9.

Tabla III.4 Tabla con los resultados obtenidos en las ejecuciones del algoritmo de alineamiento múltiple de secuencias T-Coffee utilizando como entrada secuencias sintéticas de 500 residuos.

Nodos	Tiempo Total Fase 1 (Pre-Procesamiento)	Tiempo Total Fase 2 (T-Coffee)	Consumo Memoria Fase 1 %Mem	Tiempo Total
1	29407 seg.	5281seg.	2.8	34688 seg.
2	15315 seg.	5280 seg.	2.8	20595 seg.
3	10308 seg.	3168 seg.	2.8	13476 seg.
4	7840 seg.	2370 seg.	2.8	10210 seg.
5	6460 seg.	2366 seg.	2.8	8826 seg.
6	5310 seg.	1749 seg.	2.8	7059 seg.
7	4559 seg.	1570 seg.	2.8	6129 seg.
8	4648 seg.	1570 seg.	2.8	6218 seg.

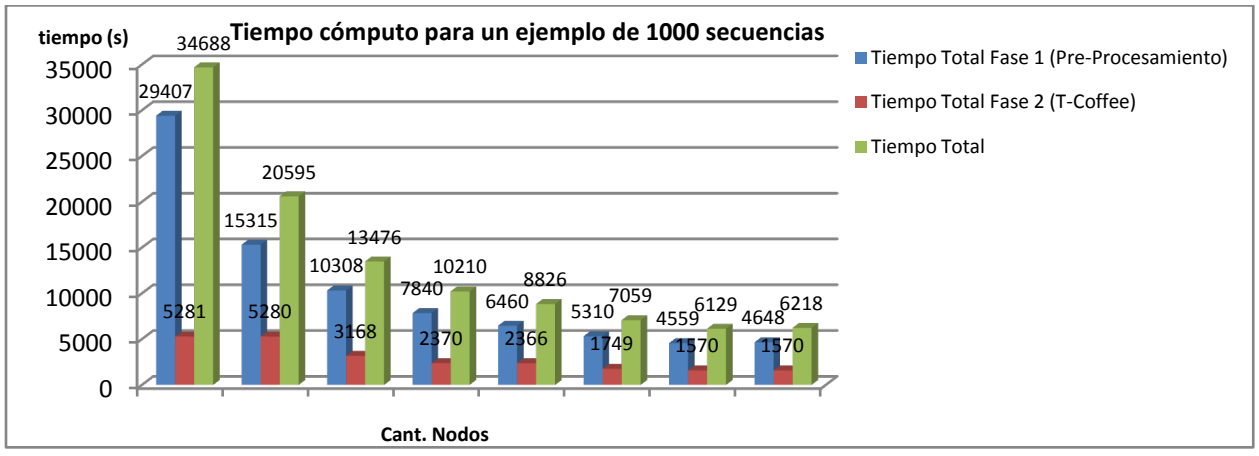


Figura III.9 Gráficos de medición de tiempo en la ejecución de un ejemplo de 1000 secuencias con diferentes cantidades de nodos de cómputo.

La Figura III.9 muestra una gráfica donde se representa en el eje de las abscisas la cantidad de nodos de cómputo en los que se realizaron los experimentos, en el eje de las ordenadas el tiempo en segundos y las barras representan el comportamiento de fases del modelo paralelo presentado en la investigación y el tiempo total, todo para el experimento de mil secuencias de entrada.

Los resultados obtenidos inicialmente muestran que es posible utilizando el mismo algoritmo T-Coffee, disminuir el tiempo de ejecución para obtener el resultado de un alineamiento de miles de secuencias en un tiempo razonablemente corto, si lo comparamos con los días que demoró el algoritmo original para luego ser terminado sin obtener ningún resultado satisfactorio.

Además los resultados muestran que en la medida que se aumenta la cantidad de nodos, el modelo escala bien, aunque se observa que la fase que consume casi todo el tiempo es la de pre-procesamiento. Esta fase depende a su vez de una serie de alineamientos entre pares de secuencias que pueden ser optimizados, pueden ejecutarse en una máquina con mayor potencia de cómputo y es una fase previa a cualquier alineamiento posterior (*en esta fase solo necesitamos los resultados de los alineamientos*).

Si analizamos los valores de tiempo de la fase de ejecución de T-Coffee (*segunda fase*) que se muestran en la Figura III.10, el modelo escala en tiempo pero lo hace de forma discreta. La explicación a este hecho es que como los grupos que se distribuyen en los nodos no tienen tamaño uniforme (*la cantidad de secuencias en cada grupo dependerá del valor de cutoff*), el comportamiento lo regirán los grupos grandes obtenidos y a que nodos le sean asignados. En la medida que existan menos nodos y a uno le correspondan dos grupos grandes, se verá afectada la escalabilidad del modelo (*la formación de cluster es con sentido biológico, no hay tamaños predefinidos o sea la granularidad es variable*). El reparto de carga equitativa puede ser un factor a analizar cuando el número de secuencias sea muy elevado (*mayor a diez mil secuencias*).

Un ejemplo de lo anterior puede ser el siguiente: supongamos que varios *clusters* pequeños con una cantidad de secuencias igual a la de uno grande y estos *clusters* pequeños sean asignados a un nodo, y el *cluster* grande de secuencias a otro; ambos nodos tendrían la misma cantidad de secuencias a procesar, pero uno de ellos tiene las secuencias distribuida en varios *clusters*; por el mismo comportamiento en el primer experimento, el nodo que tiene que ejecutar un solo *cluster*, puede demorar más, que el que tiene que realizar ejecuciones para cada uno de los *cluster* pequeños. Más secuencias a relacionar implican más tiempo de cómputo (N^2).

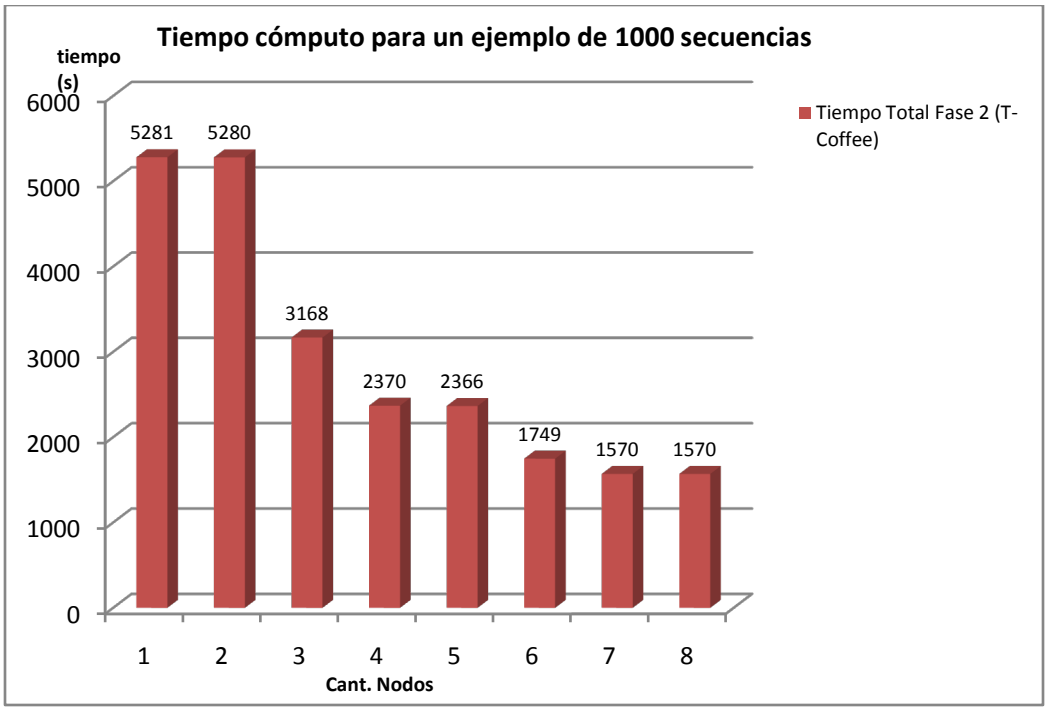


Figura III.10 Gráficos de medición de tiempo en la fase 2 del modelo para la ejecución de un ejemplo de 1000 secuencias con diferentes cantidades de nodos de cómputo.

Este experimento también nos muestra que a pesar de que una entrada de datos de miles de secuencias, forme varios grupos de secuencias, la demora estará determinada solo por los grupos más grandes (*el peor de nuestro caso grupos de 100 secuencias*), esto hará que pasado un número de nodo el modelo no escale porque el modelo se verá afectado por el nodo que más lento vaya (*el que tenga asignado el grupo mayor*). Sin embargo este comportamiento solo es para la segunda fase del algoritmo.

La primera fase escala con gran facilidad, porque como se explicó en el modelo está formado por N*(N-1)/2 tareas independientes, siendo N el número de secuencias a alinear en el alineamiento múltiple. Para mostrar esto mostremos el tiempo de la fase de pre-procesamiento de forma independiente en la Figura III.11.

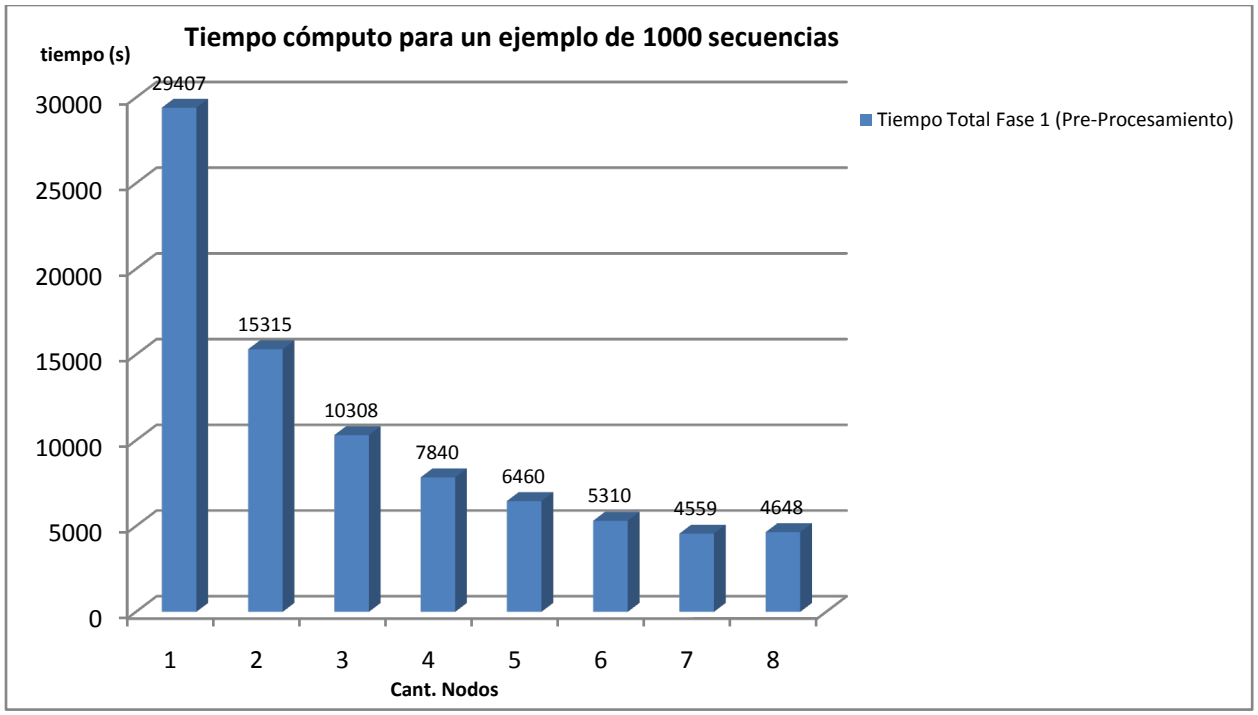


Figura III.11 Gráficos de medición de tiempo en la fase 1 del modelo para la ejecución de un ejemplo de 1000 secuencias con diferentes cantidades de nodos de cómputo.

III.4.2 Consumo de memoria principal

Para analizar el consumo de memoria, realizamos mediciones en los nodos de cómputo, para ver qué porcentaje representan los valores más altos de consumo cuando se divide en *clusters* la entrada de datos de 1000 secuencias, las mediciones se realizaron en la generación de la librería primaria y en la generación de la librería extendida. Además se realizaron mediciones del consumo de memoria en la fase de pre-procesamiento del algoritmo. Los resultados se muestran en las Figuras III.12 y III.13. Los gráficos de las Figuras III.12 y III.13 representan el valor más elevado de porcentaje de consumo de memoria en los nodos de cómputo, y se dividen en las fases del algoritmo T-Coffee cuando es ejecutado en el modelo propuesto y la fase de pre-procesamiento del modelo. En comparación con T-Coffee original, mejoramos muchísimo a primera vista, porque en principio para el mismo fichero de entrada con los que se realizaron estos experimento, se intento con T-Coffee original y al cabo de los días, el sistema termino la ejecución por consumir toda la memoria del sistema.

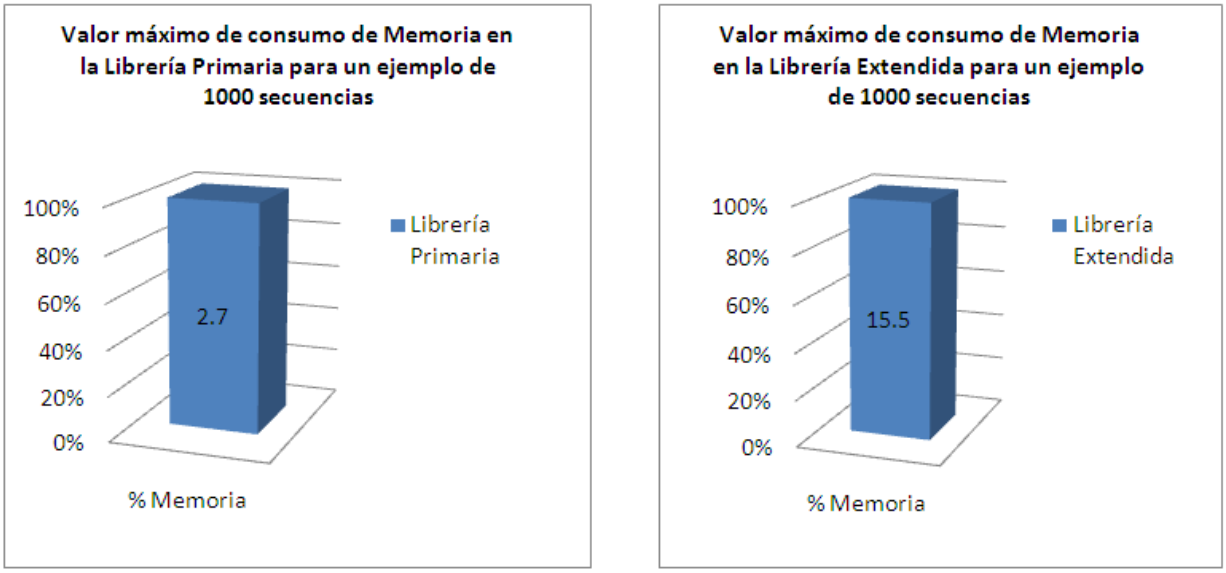


Figura III.12 Gráficos de la medición del porcentaje de consumo de memoria del modelo para la ejecución de un ejemplo de 1000 secuencias con diferentes cantidades de nodos de cómputo.

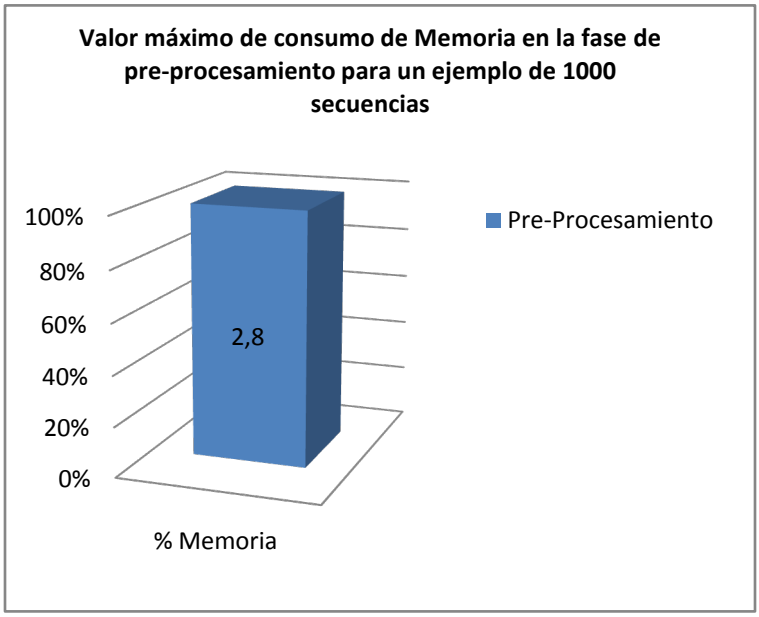


Figura III.13 Gráfico de medición del porcentaje de consumo de memoria del modelo (Fase 1) para la ejecución de un ejemplo de 1000 secuencias cómputo.

Los resultados que muestran estás gráficas nos dan un margen de alcance en la escalabilidad elevado, porque si lo más que consumimos en el entorno de experimentación es un 15.5% de consumo de memoria en la fase más crítica, hay margen de crecimiento para poder aumentar la

entrada a otros miles de secuencias de entrada. De forma general el modelo disminuye el consumo de recursos, a pesar de haber introducido una fase (*de pre-procesamiento*), que es costosa desde el punto de vista de tiempo de cómputo, pero es lineal según el número de secuencias de entrada y con margen de optimización. La comparación con T-Coffee original es muy ventajosa, porque se ha disminuido mucho el consumo de recursos, al punto que el modelo solamente con la separación de grupos de secuencias, logra resolver el problema de saturación al analizar 1000 secuencias en un tiempo muy razonable en comparación con el original.

III.5 Análisis final de ClusT-Coffee.

Luego de haber realizado el grupo de experimentos antes expuesto podemos asegurar que el modelo propuesto para T-Coffee mejora en gran medida el algoritmo original. Se mostró que la fase que se introduce es independiente al algoritmo de alineamiento y que aunque emplee los mismos métodos no interfiere en el resultado de T-Coffee. Esta fase introducida es la que nos permite clusterizar las secuencias de entrada, de forma que luego el algoritmo disminuya el consumo de recursos y el tiempo de ejecución al ejecutarlas en paralelo. Los resultados obtenidos en los experimentos de calidad de alineamientos, mostraron que hay margen de mejora, y que es necesario crear un esquema de evaluación para que en dependencia de las características (*porcentaje de similitud*) de las secuencias de entrada, definir como se clusterizan las secuencias.

CAPITULO IV

Conclusiones y líneas abiertas

En este capítulo se presentarán las conclusiones de la investigación desarrollada y las líneas abiertas que quedan en el problema de alineamiento múltiple de secuencias. Las aplicaciones de alineamiento múltiple de secuencias son prototipos de aplicaciones que requieren elevada potencia de cómputo y memoria. Se destacan por la relevancia científica que tienen los resultados que brindan a investigaciones científicas en el campo de la biomedicina, genética y farmacología, y se destacan además por la complejidad y el reto que representan computacionalmente.

Los algoritmos empleados en este tipo de aproximaciones intentan, extraer conocimientos de los datos que analizan, por lo que se hace mucho énfasis en la precisión y exactitud de los algoritmos. Como se comento al inicio del documento, hoy día aún no es posible obtener la solución óptima a este problema, y todos los algoritmos se concentran en extraer de los datos toda la información posible, que resulta en un aumento de la complejidad del problema y limita en la mayoría de los casos la cantidad de datos a analizar.

Hoy por hoy, los avances en los estudios genéticos están proporcionando volúmenes de datos inmensamente elevados, creándose repositorios de información que implícitamente contemplan un conocimiento histórico de nuestra evolución. Estos datos necesitan ser procesado y analizado, y se ha hecho evidente que el único entorno para darle solución al problema es el de Computación de Altas Prestaciones.

Dentro de los algoritmos de alineamiento múltiples de secuencias se encuentra T-Coffee, que ha sido desarrollado en el Centro de Regulación Genómica de Barcelona (*CRG*) y es utilizado en estudios e investigación de enfermedades como el cáncer. Este algoritmo es el mejor dentro de su clasificación, pero tiene como limitante la cantidad de secuencias que puede procesar, principalmente por el consumo de memoria y tiempo de cómputo. El algoritmo no puede procesar miles de secuencias, y esto es un freno hoy día.

La investigación realizada por nosotros se enfocó en crear un modelo que le permitiese a los algoritmos de alineamiento múltiple de secuencias, aumentar el número de secuencias a procesar tratando de mantener la calidad en los resultados, para garantizar la precisión científica y la posible aplicación de cualquier conocimiento extraído de los datos. Para presentar y validar el modelo utilizamos a T-Coffee, como algoritmo de referencia e investigación. Pero el modelo puede migrarse a cualquier otro algoritmo.

IV.I Conclusiones

El objetivo central de la investigación fue crear un modelo para paralelizar T-Coffee en particular los algoritmos de alineamiento múltiple de secuencias en general, que permitiese aumentar el número de datos de entrada obteniendo la misma calidad en los resultados del propio algoritmo y disminuyendo a la vez el tiempo de cómputo. Para diseñar el modelo, se estudiaron los algoritmos de alineamiento múltiple de secuencias, para poder enfocar el modelo de forma general, aunque se utilizase T-Coffee como algoritmo de referencia. En el caso del algoritmo T-Coffee, estudiamos sus fases principales, para poder analizar el mecanismo de procesamiento de los datos y la complejidad de cada fase. Esta tarea nos permitió entender por qué tenía problemas de consumo de memoria y cuál era la fase crítica, los detalles son:

- ✓ T-Coffee utiliza un algoritmo que de forma general presenta tres fases: librería primaria (*alineamientos en pares para todas las secuencias de entrada que tiene un grado de complejidad elevado, pero no es determinante por el momento en el método*), librería extendida (*búsqueda de relaciones entre todas las secuencias de entrada a través de pares de residuos, complejidad muy elevada*), alineamiento múltiple progresivo (*fase de construcción de un árbol guía que luego expresar las relaciones en el alineamiento progresivo, complejidad baja en comparación con las fases anteriores*). Esto permite optimizar el proceso por fases.
- ✓ La construcción de la librería primaria es una fase que influye mucho en el tiempo de cómputo, es donde se obtienen los alineamientos en pares para todas las secuencias de entrada, tiene una complejidad elevada (*aumenta de forma cuadrática con el número de secuencias de entrada*). Tiene procesos que reservan mucha memoria de cómputo para almacenar información de las relaciones entre pares de residuos alineados en todos los

alineamientos en pares. Para pocos cientos de secuencia no es crítico, pero para miles es un proceso muy costoso.

- ✓ La construcción de la librería extendida es la fase limitante de la aplicación, se demostró experimentalmente que consume mucha memoria, con muy pocas secuencias de entrada. Todo el proceso se debe a que reserva mucha memoria para relacionar pares de residuos a través de un tercero. Esta fase lo que intenta es extraer el conocimiento implícito de las secuencias de entrada y es la fase crítica de consumo de recursos de cómputo que necesitan minimizarse, intentando amortiguar la extracción de conocimientos limitándose solo a las secuencias que brinden información útil. Para miles de secuencias es casi imposible obtener una librería extendida, por la cantidad de memoria que puede demandar.
- ✓ La última fase es la que determina la calidad del alineamiento, la complejidad no es tan elevada como en las fases anteriores. En esta fase se debe tratar de mantener la calidad en todo momento, aunque aumente la cantidad de secuencias a procesar a miles. Su exactitud está completamente acoplada a la fase de construcción de la librería extendida, que es la que complementa el alineamiento con la información implícita en las secuencias de entrada.

Como resultado de la investigación y el estudio de las fases del algoritmo (*prototipo de algoritmos progresivos basados en consistencia*) presentamos un modelo que se basa en la formación de clusters de secuencias a partir de las secuencias de entrada como primera etapa, para obtener en una segunda etapa alineamientos múltiples de cada uno de los grupos de secuencias formados, que luego serán empleados por el algoritmo para obtener el alineamiento final. Este último alineamiento emplearía cada alineamiento múltiple de los grupos como si fuesen una secuencia única, reduciendo en todo momento la complejidad del problema a tratar. El modelo es independiente del algoritmo de alineamiento a utilizar y tiene la ventaja de que cada uno de los alineamientos *intra-cluster* que se calculen se puedan lanzar en paralelo. Y que la fase inicial de alineamientos en pares también puede paralelizarse por la cantidad de tareas independientes que lo forman.

De los resultados obtenidos en la experimentación concluimos que el modelo es factible, disminuye el consumo de tiempo de cómputo y memoria en todo momento. Sí se observó, que tanto la calidad, como el tiempo de cómputo y consumo de memoria son dependientes de la forma en que se considera que dos secuencias de entrada pertenecen a un mismo *cluster*, y esto

lo determina el límite de similitud que usamos en secuencias que se deseen agrupar bajo cada grupo. Como conclusión de este análisis, creemos que se debe diseñar un modelo que sea capaz de como parámetro determinar cuál puede ser el valor de similitud ideal o mejor, en dependencia de las características de los datos de entrada. Este modelo de similitud puede buscar una relación entre tamaño de cluster y valor de *cutoff* pero siempre debe tener sentido biológico.

En resumen podemos asegurar que se han obtenidos resultados satisfactorios por:

- ✓ Haber creado un modelo para paralelizar los algoritmos de alineamiento múltiple de secuencias, que permite aumentar el número de datos de entrada obteniendo la misma calidad en los resultados, disminuyendo a la vez el tiempo de cómputo y consumo de memoria. El modelo se probó empleando como caso de estudio T-Coffee. Este modelo es extensible a otros algoritmos porque no se hace dependiente en ningún momento del caso de estudio empleado T-Coffee, ya que es una etapa previa e independiente al alineamiento.
- ✓ Haber validado el modelo con experimentos que demuestran su exactitud biológica. Y haber solucionado el problema de tiempo de ejecución y consumo de memoria, demostrado de forma experimental.

IV.II Líneas abiertas

A pesar de haber obtenido un modelo que cumpliese los objetivos de la investigación, queda mucho que hacer en esta área. Como líneas abiertas en la investigación, para seguir avanzando en la solución de la problemática de alineamientos múltiples de secuencia queda:

- ✓ Para el modelo propuesto:
    - o Crear un algoritmo paralelo eficiente para obtener los grupos de secuencias que más se asemejan a partir un número de secuencias de entrada.
    - o Considerar una forma dinámica de asignar las tareas (*asignación de grupos de secuencias a alinear*) a nodos de cómputo.
    - o Implementar alguna estrategia de tolerancia a fallos en la aplicación, para contemplar la perdida de datos de los grupos de secuencias analizados durante el proceso.

- Incidir en las fases del algoritmo, que pueden ser paralelizadas, ya que en la investigación se utilizó el paralelismo de cara a los datos (*descomposición de datos*), pero no de cara a las etapas funcionales del algoritmo.
- Generar un algoritmo que permita analizar el comportamiento de los datos y definir cuál es el mejor valor de similitud entre los datos para formar los grupos, considerando específicamente la entrada de datos.

✓ Migrar el modelo utilizando como caso de estudio un algoritmo de otra categoría como los métodos de alineamiento iterativo.

✓ Realizar alineamientos de secuencias más allá de las 1000 secuencias.

✓ Integrar el modelo como módulo optimizado de T-Coffee en el servidor del CRG para ponerlo a disposición de toda la comunidad científica.

Todos estos puntos son sensibles a investigaciones futuras que pueden contribuir en la solución del problema de alineamiento múltiple de secuencias.

CAPITULO V

Bibliografía

- Agrawal, A., Khaitan, S.K. (2008) A new heuristic for multiple sequence alignment. Electro/Information Technology, 2008. EIT 2008. IEEE International Conference on 2008:215-217
- Athas, W. C. & Seitz, C.L. (1998). "Multicomputers: Message-Passing Concurrent Computers" IEEE Computer, v.21, n.8 pp.9-24.
- Arslan, A.N., Bizargity, P. (2007) Phylogeny By Top Down Clustering Using a Given Multiple Alignment. Bioinformatics and Bioengineering, 2007. BIBE 2007. Proceedings of the 7th IEEE International Conference on 2007:809-814.
- Bilu, Y., Agarwal, P.K., Kolodny, R. (2006). Faster Algorithms for Optimal Multiple Sequence Alignment Based on Pairwise Comparisons. Computational Biology and Bioinformatics, IEEE/ACM Transactions on 2006;3(4):408-422.
- Carrilo, H. & Lipman, D. J. (1988). The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.* 48, 1073-1082.
- Chakrabarti, S., Lanczycki, C., Panchenko, A., Przytycka, T., Thiessen, P., Bryant, S. (2006). "State of the Art: Refinement of Multiple Sequence Alignments." BMC Bioinformatics 7.1 (2006): 499.
- Culler, D.E. & Singh, J.P. (1999). "Parallel Computer Architecture. A Hardware/Software Approach". Morgan Kaufmann Pub.
- Deng, X., Li, E., Shan, J., Chen, W. (2006). Parallel implementation and performance characterization of MUSCLE. Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International 2006:7 pp
- Finn, J., Tate, J., Mistry, P.C., Coggill, J.S., Sammut, H.R., Hotz, G., Ceric, K., Forslund, S.R., Eddy, E.L., Sonnhammer and A. Bateman. (2008) The Pfam protein families database: R.D. Nucleic Acids Research. Database Issue 36:D281-D288
- Flynn, M. J. (1972). Some Computer Organisations and their effectiveness. IEEE Transaction on Computers, c-21(9) pp. 114-118.

- Franco, D. (2008). “Balanceo distribuido del encaminamiento en redes de interconexión de computadores paralelos”. Departament d’Informàtica, Universitat Autónoma de Barcelona.
- Geist, A., Beguelin, A., Dongarra, J., Jiang, W. Manchek, R. Sunderam, V. (1994), “Parallel Virtual Machine. A User’s Guide and Tutorial for Networked Parallel Computing”. Oak Ridge National Laboratory.
- Helal, M., El-Gindy, H., Mullin, L., Gaeta, B. (2008). Parallelizing Optimal Multiple Sequence Alignment by Dynamic Programming. Parallel and Distributed Processing with Applications, 2008. ISPA '08. International Symposium on 2008:669-674.
- Hongwei, H., Stojkovic, V. (2007). A simulated annealing algorithm for multiple sequence alignment with guaranteed accuracy. Natural Computation, 2007. ICNC 2007. Third International Conference on 2007;2:270-274.
- Hwang, K. (1993). “Advanced Computer Architecture: Parallelism, Scalability, Programmability”. Mc Graw-Hill.
- Kececioglu, J. D. (1993). The maximum weight trace problem in multiple sequence alignment. *Lect. Notes Comput. Sci.* **684**, 106-119.
- Kupis, P., Mandziuk, J. (2007). Evolutionary-Progressive Method for Multiple Sequence Alignment. Computational Intelligence and Bioinformatics and Computational Biology, 2007. CIBCB '07. IEEE Symposium on 2007:291-297.
- Masuno, S., Maruyama, T., Yamaguchi, Y., Konagaya, A. (2007). An FPGA Implementation of Multiple Sequence Alignment Based on Carrillo-Lipman Method. Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on 2007:489-492.
- Message Passing Interface Forum. (1994). “MPI. A Message-Passing Interface Standard”. International Journal of Supercomputer Application and High Performance Computing, v.9, no.3/4.
- Needleman, S.B., Wunsch, C.D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins". *J Mol Biol* **48** (3): 443–53.
- Notredame, C., Higgins, D. G., Heringa, J. (2000) "T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment." J.Mol.Biol. 302: 205-17.
- Rezaei, S. & Monwar, Md. M. (2006). Divide-and-Conquer Algorithm for Clustalw-MPI. Electrical and Computer Engineering, 2006. CCECE '06. Canadian Conference on 2006:717-720.

- Rezaei, S, Monwar, M.M., Bai, J. (2006). Performance Comparison of MPI-Based Parallel Multiple Sequence Alignment Algorithm Using Single and Multiple Guide Trees. Cognitive Informatics, 2006. ICCI 2006. 5th IEEE International Conference on 2006;1:595-600.
- Saeed, F. & Khokhar, A. (2008). Sample-Align-D: A high performance Multiple Sequence Alignment system using phylogenetic sampling and domain decomposition. Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on 2008:1-9.
- Saito, N. & Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* **4**, 406-125.
- Sande, C. & Schneider, R. (1991). Database of homology-derived protein structures and structural meaning of sequence alignment. *Proteins: Struct. Funct. Genet.* 9, 56-68.
- Seitz, C. L. (1990). "Multicomputers. Developments of Concurrency and Communication" Addison-Wesley, pp.131-200.
- Smith, T.F., Waterman, M.S. (1981). Identification of common molecular subsequences. *J Mol Biol.* **147** (1): 195-7.
- Sokal, R.R. & Michener, C.D. (1958). A statistical method for evaluating systematic relationships. *Univ. Kansas Sci. Bull*. **38**: 1409-1438
- Suaya, R. & Birtwistle, G. (1990). VLSI And Parallel Computation Frontiers. Morgan Kaufmann Publishers.
- Thompson, J.D., Koehl, R., Ripp, R., Poch, O. (2005). BAliBASE 3.0: Latest Developments of the Multiple Sequence Alignment Benchmark. PROTEINS: Structure, Function, and Bioinformatics 61:127–136.
- Wilkinson, B. (1996). "Computer Architecture. Design and Performance", 2d. Edition, Prentice-Hall.
- Zola, J., Trystram, D., Tchernykh, A., Brizuela, C. (2006). Parallel multiple sequence alignment with local phylogeny search by simulated annealing. Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International 2006:8 pp.
- Zola, J., Yang, X., Rospondek, S., Aluru, S. (2007). *Parallel T-Coffee: A Parallel Multiple Sequence Aligner*, In Proc. of ISCA PDCS-2007, pp. 248-253.

ANEXOS

Anexo I

Base de datos Balibase 3.0 (*benchmark*)

La base de datos BALIBASE (del inglés **B**enchmark **ALI**gnment data**BASE**), fue diseñada para servir como evaluadora de los algoritmo de alineamiento de múltiple de secuencias. La base de datos presenta una elevada calidad en los datos de las secuencias, que han sido construidos manualmente. La base de datos está formada por familias de proteínas (*proteínas con la misma función biológica*) y se encuentra clasificada según características que identifican a las familias como la similitud entre las proteínas que la conforman. La clasificación es la que se muestra a continuación:

Referencias

Reference 1 equi-distant sequences with 2 different levels of conservation,

Reference 2 families aligned with a highly divergent "orphan" sequence,

Reference 3 subgroups with <25% residue identity between groups,

Reference 4 sequences with N/C-terminal extensions,

Reference 5 internal insertions.

Identificadores de las referencias anteriores

RV11: Reference 1, very divergent sequences (<20% identity)

RV12: Reference 1, medium to divergent sequences (20-40% identity)

RV20: Reference 2

RV30: Reference 3

RV40: Reference 4

RV50: Reference 5

La base de datos Balibase se encuentra disponible para la comunidad científica en la URL (http://www-bio3d-igbmc.u-strasbg.fr/balibase/). Entre los datos presente en Balibase se encuentra, varios ficheros textos por cada categoría con familias de proteínas según la categoría correspondiente, y además dispone de un fichero texto con el resultado del alineamiento para las secuencias de la familia de proteína que propone. Balibase dispone un programa que permite comparar el alineamiento correcto presente en la base de datos, con otro obtenido por algún método de alineamiento múltiple de secuencias, con el fin de verificar la calidad del alineamiento obtenido por el método.

Resultados Obtenidos en una ejecución para 8 Nodos

Resultados obtenidos para la ejecución presentada en la experimentación con 1000 secuencias (*con un valor de cutoff de 35% de indetidad de secuencia para la generación de los cluster*), generándose 66 *clusters*. Los tiempos en segundo fueron redondeados a dos valores decimales, lo que provocó que se obtuviesen ceros en algunos tiempos.

A continuación se muestra un grupo de Tablas con los resultados de los tiempos medidos en la fase de experimentación para 1000 secuencias. Los promedios de los tiempos fueron redondeados a dos cifras decimales, lo que produjo que algunos valores fuese de cero. Los valores de tiempo igual a cero se mantuvieron porque lo que representan son tiempos de cómputo despreciables frente a casos dentro del mismo ejemplo.

Tabla A.1 Tabla con los resultados de tiempo (promedio) en el nodo *master* para la ejecución de la fase 2 del modelo ClusT-Coffee, con la entrada de 1000 secuencias utilizada en la experimentación.

Nodo	Tiempos (Segundos)					
	Inicialización	Lib. Primaria	Lib. Extendida	Construcción Árbol	Alineamiento	Total
1 (master)	2	296	255	0	41	594

Tabla A.2 Tabla con los resultados de tiempo (promedio) en el nodo 2 para la ejecución de la fase 2 del modelo ClusT-Coffee, con la entrada de 1000 secuencias utilizada en la experimentación.

Nodo	Tiempos (Segundos)					
	Inicialización	Lib. Primaria	Lib. Extendida	Construcción Árbol	Alineamiento	Total
2	3	375	299	0	115	792
	0	0	0	0	0	0
	0	1	0	0	0	1
	0	0	0	0	0	0
	0	1	2	0	1	4
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	1	0	0	1
Total	3	377	302	0	116	798

Tabla A.3 Tabla con los resultados de tiempo (promedio) en el nodo 3 para la ejecución de la fase 2 del modelo ClusT-Coffee, con la entrada de 1000 secuencias utilizada en la experimentación.

Nodo	Tiempos (Segundos)					
	Inicialización	Lib. Primaria	Lib. Extendida	Construcción Árbol	Alineamiento	Total
3	3	375	299	0	115	792
	0	1	2	0	0	3
	0	1	0	0	0	1
	0	0	0	0	0	0
	0	0	1	0	0	1
	0	1	2	0	1	4
	0	0	0	0	0	0
	0	1	0	0	0	1
	0	0	0	0	0	0
Total	3	379	304	0	116	802

Tabla A.4 Tabla con los resultados de tiempo (promedio) en el nodo 4 para la ejecución de la fase 2 del modelo ClusT-Coffee, con la entrada de 1000 secuencias utilizada en la experimentación.

Nodo	Tiempos (Segundos)					
	Inicialización	Lib. Primaria	Lib. Extendida	Construcción Árbol	Alineamiento	Total
4	1	375	299	0	115	790
	0	1	1	0	0	2
	0	0	0	0	1	1
	0	0	0	0	0	0
	0	0	1	0	0	1
	0	0	1	0	0	1
	0	0	0	0	0	0
	0	0	0	0	0	0
Total	1	376	302	0	116	795

Tabla A.5 Tabla con los resultados de tiempo (promedio) en el nodo 5 para la ejecución de la fase 2 del modelo ClusT-Coffee, con la entrada de 1000 secuencias utilizada en la experimentación.

Nodo	Tiempos (Segundos)					
	Inicialización	Lib. Primaria	Lib. Extendida	Construcción Árbol	Alineamiento	Total
5	2	375	299	0	115	791
	0	0	0	0	0	0
	0	1	0	0	0	1
	0	1	1	0	0	2
	0	1	1	0	0	2
	0	1	1	0	0	2
	0	0	0	0	0	0
	0	0	0	0	0	0
Total	2	379	302	0	115	798

Tabla A.6 Tabla con los resultados de tiempo (promedio) en el nodo 6 para la ejecución de la fase 2 del modelo ClusT-Coffee, con la entrada de 1000 secuencias utilizada en la experimentación.

Nodo	Tiempos (Segundos)					
	Inicialización	Lib. Primaria	Lib. Extendida	Construcción Árbol	Alineamiento	Total
6	2	442	423	0	97	964
	0	1	0	0	1	2
	0	1	4	0	0	5
	0	1	0	0	0	1
	0	0	1	0	0	1
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	1	0	0	1
Total	2	445	429	0	98	974

Tabla A.7 Tabla con los resultados de tiempo (promedio) en el nodo 7 para la ejecución de la fase 2 del modelo ClusT-Coffee, con la entrada de 1000 secuencias utilizada en la experimentación.

Nodo	Tiempos (Segundos)					
	Inicialización	Lib. Primaria	Lib. Extendida	Construcción Árbol	Alineamiento	Total
7	2	131	160	0	66	359
	0	0	1	0	0	1
	0	5	11	0	1	17
	0	1	2	0	0	3
	0	1	0	0	0	1
	0	0	0	0	0	0
	0	1	0	0	0	1
	0	0	0	0	0	0
	0	0	1	0	0	1
	0	0	0	0	0	0
Total	2	139	175	0	67	383

Tabla A.8 Tabla con los resultados de tiempo (promedio) en el nodo 8 para la ejecución de la fase 2 del modelo ClusT-Coffee, con la entrada de 1000 secuencias utilizada en la experimentación.

Nodo	Tiempos (Segundos)					
	Inicialización	Lib. Primaria	Lib. Extendida	Construcción Árbol	Alineamiento	Total
8	3	1	4	0	1	9
	0	29	56	0	5	90
	0	2	4	0	1	7
	0	0	2	0	0	2
	0	0	1	0	0	1
	0	4	9	0	1	14
	0	1	1	0	0	2
	0	0	0	0	0	0
	0	1	0	0	0	1
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
Total	3	38	77	0	8	126

Tabla A.9 Tabla con los resultados de tiempo (promedio) de la ejecución en 8 nodos del modelo ClusT-Coffee, con la entrada de 1000 secuencias utilizada en la experimentación.

Nodos	Tiempo Total Fase 1 (Pre-Procesamiento)	Tiempo Total Fase 2 (T-Coffee)	Consumo Memoria Fase 1			Tiempo Total
			RES	Virtual	%Mem	
8	4648	1570	27	32	2,8	6218

Tabla A.10 Tabla con la cantidad de clusters asignados a los nodos de cómputo en la ejecución en 8 nodos del modelo ClusT-Coffee, con la entrada de 1000 secuencias utilizada en la experimentación.

Nodos	Tiempo Total de Cómputo (Fase 2)	Cantidad de Clusters
1	594	0
2	798	9
3	802	9
4	795	8
5	798	8
6	974	9
7	383	10
8	126	13
Total	5270	66

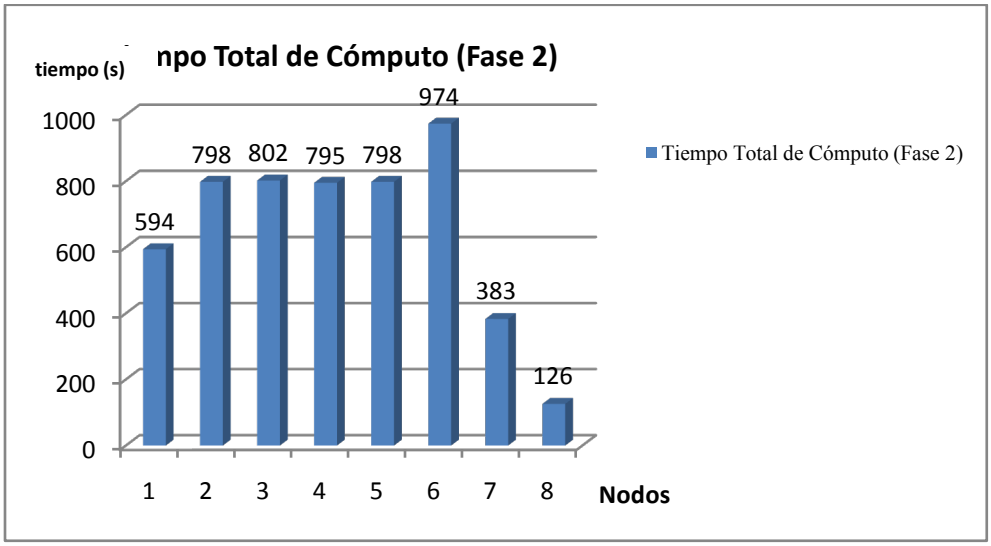


Figura A.1 Gráfico con los tiempos de cómputo en cada uno de los nodos, para la obtención de los alineamientos de los clusters, en la ejecución del modelo ClusT-Coffee del ejemplo de 1000 secuencias del experimento.

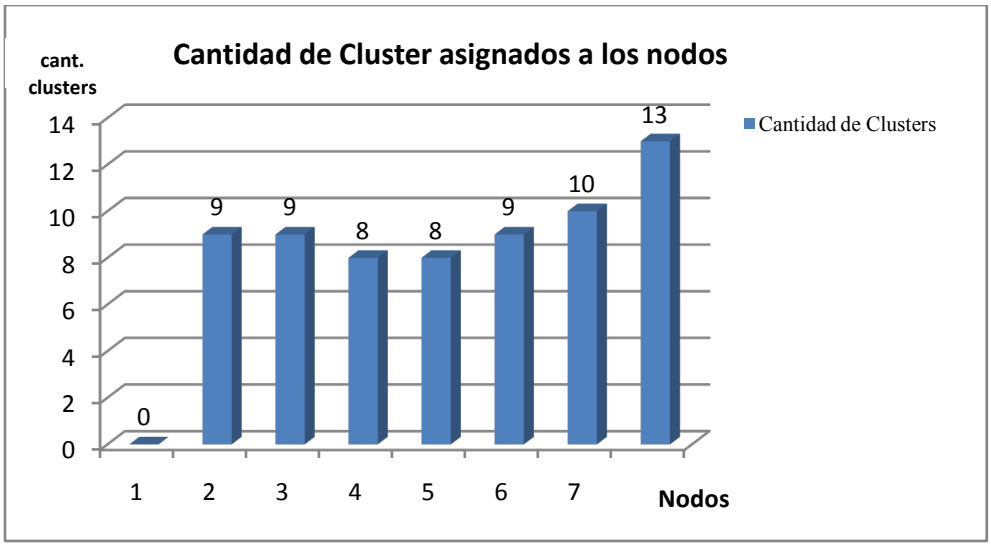


Figura A.1 Gráfico con los *cluster asignados* a cada uno de los nodos, para la obtención de los alineamientos de los clusters, en la ejecución del modelo ClusT-Coffee del ejemplo de 1000 secuencias del experimento.