BÚSQUEDA DE SECUENCIAS REPETIDAS EN UN MODELO DE SECUENCIAMIENTO ALEATORIO.

Alumno: Cristian S. Rocha (LU 369/96)
Directora: Lic. Irene Loiseau

Tesis de Licenciatura

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Agosto de 2002

Resumen

El objetivo de esta tesis es proponer una nueva metodología para identificar secuencias repetidas exactas en un conjunto de fragmentos de una secuencia de ADN, y así permitir el acceso a estos elementos durante su proceso de secuenciamiento aleatorio. Surgido de la limitación de conocer el genoma completo del *Trypanosoma cruzi*, para luego así obtener todas sus repeticiones para su eventual anotación, este trabajo introduce la teoría de *álgebra de repeticiones*. Permitiendo así relacionar correctamente una nueva estructura de datos, AS^2, con el problema de encontrar las repeticiones supermaximales de un conjunto de subsecuencias de una secuencia.

Abstract

These work introduces a new method to identify exact DNA repeats from a set of fragments of a DNA sequence, to allow queries about them while the random sequence strategy is in progress. The limits to know the complete genome of *Trypanosoma cruzi*, do not permit to researchers to annotate all repeats in the known way. In order to solve this problem, these thesis introduces the *repeats algebra* theory to relate a new data structure, AS^2, to the *supermaximal repeats in a set of subsequences of a sequence search problem*.

Dedico este esfuerzo a la memoria de mi padre; y a mi hermano que fue a probar suerte fuera de en otro país.

Agradecimientos.

Antes las eventuales idas y vueltas del desarrollo de esta tesis me siento comprometido a agradecer a una lista grande de personas a quienes tengo un gran afecto.

Agradezco a Gabriel Tamburelli, Juan H. Ortega, Maximiliano Sacco y Alejandro Gomez por su apoyo logístico. A mi directora de Tesis, Irene Loiseau, por la confianza depositada en mi.

Pero el apoyo incondicional otorgado por mi madre, y Gabriela, mi compañera de vida, es el fundamento de todo mi trabajo.

Índice general

Índice de figuras

Índice de cuadros

CAPíTULO 1

Introducción

1.1. Origen de la Tesis

A mediados del año 1998 se realizó el primer acercamiento de investigadores del *Instituto de Investigaciones en Ingeniería Genética y Biología Molecular* (INGEBI) con investigadores del *Departamento de Computación* (DC), ambos pertenecientes de la *Facultad de Ciencias Exactas y Naturales* (FCEN) de la *Universidad de Buenos Aires* (UBA), representados por el Dr. Mariano Levin y la Lic. Irene Loiseau respectivamente. En el mismo, se planteó la posibilidad de construir herramientas computacionales para resolver problemas específicos a tareas de investigación del INGEBI. En este primer acercamiento surgieron problemas de comunicación entre los investigadores de las dos especialidades por el desconocimiento mutuo de los términos usados, por ello se conformó el primer grupo interdisciplinario entre estudiantes, docentes e investigadores de Biología y Computación.

Este grupo tiene como primer objetivo formar especialistas en el campo de la *Biología Computacional*; y luego, pero no menos importante, construir nuevas herramientas para optimizar las tareas de investigación en *Biología Molecular*. Ambos objetivos se están llevando a cabo mediante la materia de grado *Introducción a la Biología Computacional* que se dicta en el DC, y con la implementación de *la base de datos de secuencias repetidas* del genoma del *Trypanosoma cruzi* , parásito causante del Mal de Chagas. La misma puede consultarse desde la Internet en la dirección http://machi.exp.dc.uba.ar/.

La base de datos se presentó como una solución parcial a un problema planteado por el Dr. Levin: "clasificar las secuencias repetidas del *T. cruzi*"[1]. Como se verá con más detalle en la sección 1.5, las secuencias repetidas, o repeticiones, son clases de secuencias de ADN "parecidas" que se encuentran en un mismo genoma. El INGEBI, como participante del grupo de instituciones involucradas con los proyectos genoma de

[1]Una repetición es una secuencia de símbolos que se repite dentro de otra secuencia de símbolos. Por ejemplo, la secuencia de símbolos del alfabeto "issi" es repetición de la secuencia "mississippi".

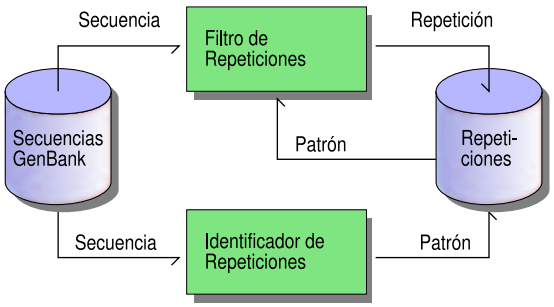

FIGURA 1.1.1. Interacción entre programas de búsqueda de repeticiones conocidas y de desconocidas. Los datos para buscar y encontrar las nuevas secuencias provienen de una *Base de Datos de Secuencias de ADN* (GenBank por ejemplo) y se las procesa para encontrar los patrones conocidos. Los mismos son almacenados en la *Base de Datos de Repeticiones* para luego ser usados por el programa de búsqueda de repeticiones existentes.

las especies Trypanosómicas - *Tri-Tryp Consortium* - se comprometió a especializarse en las repeticiones de los parásitos que se congregan en el proyecto; para ello debe recolectar, clasificar y crear una nomenclatura clara, sin ambigüedades, para cada repetición. A su vez el grupo de Biología Computacional diseñara las herramientas necesarias para completar este trabajo.

Durante el proceso de maduración del grupo surgieron dos aproximaciones diferentes para encarar el trabajo:

Búsqueda de repeticiones conocidas: El objetivo es encontrar copias de repeticiones previamente estudiadas por métodos biológicos o computacionales.

Búsqueda de nuevas repeticiones: El objetivo es encontrar patrones que indiquen la existencia de repeticiones.

Estas aproximaciones no son por completo independientes sino también complementarias, ya que la primera puede usar a la segunda como entrada de datos. En la figura 1.1.1 se puede observar la interacción entre los procesos de búsqueda de repeticiones, conocidas y nuevas. Aunque este fue el primer diseño para encarar el proyecto propuesto por el Dr. Levin, se decidió por uno que acompañe el proceso de secuenciamiento del *T. cruzi*, ver sección 1.3. Es por eso que actualmente se esta desarrollando el modelo descripto en la figura1.1.2.

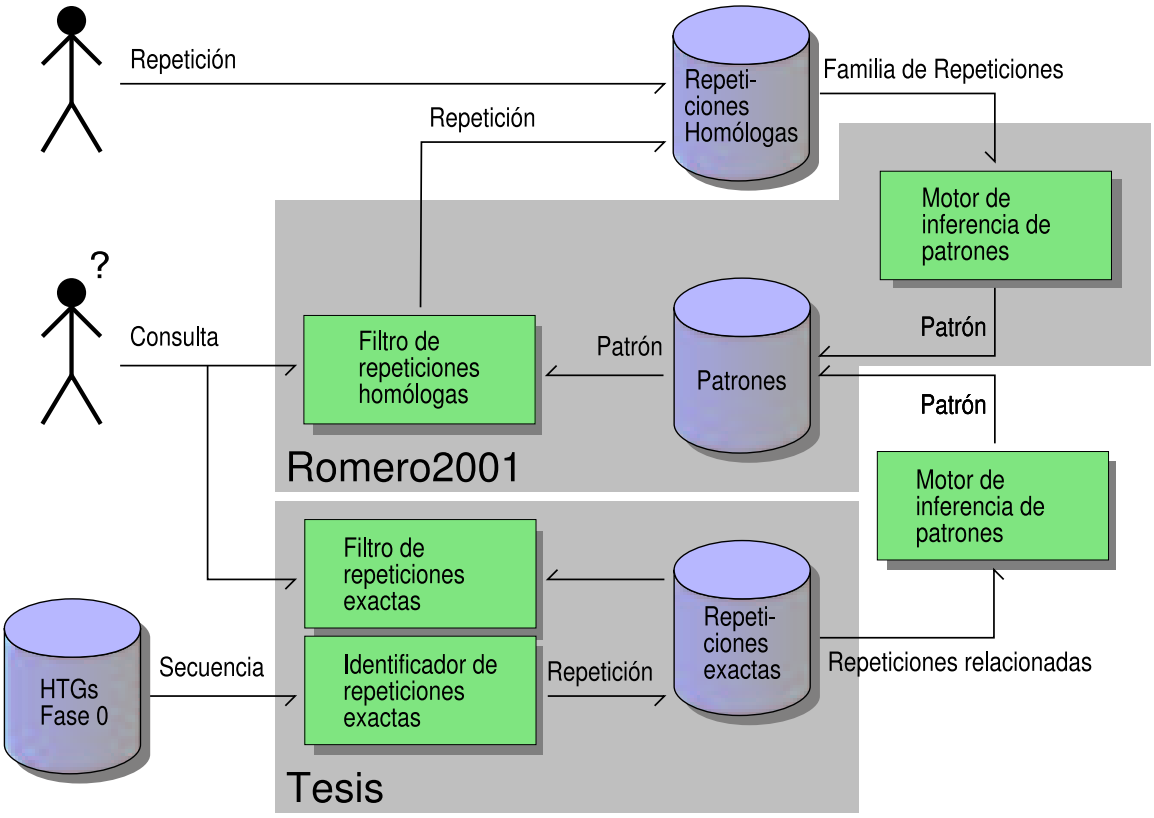

FIGURA 1.1.2. Nueva aproximación de la Base de Datos Activa. Existen dos sistemas interactuando para la búsqueda de repeticiones. El primero basado en técnicas de aprendizaje automático puede reconocer estructuras complejas de estructuras de ADN (Romero2001, ver [**38**]). El segundo sistema, basado en técnicas de combinatoria, recoge la información de una base de datos HTGs (High Throughput Genomic sequences - Secuencias genómicas obtenidas con procesos de alto rendimiento) en fase 0 (ver *The HTG GenBank Division*, http://www.ncbi.nlm.nih.gov/HTGS/) para ser procesadas en búsqueda de nuevas repeticiones exactas (Tesis). El motor de inferencia de patrones de repeticiones exactas es un modulo es una tarea pendiente del grupo interdisciplinario.

El problema principal que se encontró a la hora de encarar la búsqueda de las copias dentro de fragmentos es la necesidad de modelar la repetición, concepto biológico difuso, y es por ello que se encaró la primera aproximación con técnicas de aprendizaje automático descripto en [**38**]. En la figura 1.1.1 se muestra en gris oscuro en que parte del proyecto esta involucrado este trabajo.

Por el otro lado, en esta tesis se encara el problema de buscar nuevas repeticiones, pero no de la manera clásica: repeticiones de un fragmento. El autor plantea buscar repeticiones en un conjunto de fragmentos obtenidos aleatoriamente de una secuencia mayor: el genoma, cromosoma o fragmento. Es así que la tesis cumple con el requisito de buscar repeticiones exactas dentro de una base de datos generada a través de un secuenciamiento aleatorio. La figura 1.1.2 muestra en color gris claro los temas involucrados en esta tesis.

El trabajo esta dividido en cuatro capítulos: la presente introducción al problema de las repeticiones, como se encara este problema en la actualidad, la estructura de datos propuesta: Árbol de sufijos de supermaximales AS^2, y los resultados y conclusiones pertinentes.

Introducción: La sección 1.2, complementa los fundamentos de este trabajo, abarcando otros puntos de interés más allá del proyecto genoma del *T. cruzi*. En la siguiente, sección 1.3, se explica como las repeticiones influyen en los proyectos genomas en general. Se compara cada aproximación a los proyectos genomas en la sección 1.4. En la sección 1.5 se presenta la definición biológica de las repeticiones para luego empezar a describir el problema computacionalmente. La sección 1.6 introduce a conceptos que serán usados en los próximos capítulos.

Estado del arte: Este capítulo detalla las diferentes aproximaciones que existen en la búsqueda de repeticiones. Se intenta asociar cada aproximación al problema propuesto para luego concluir en la necesidad de construir un modelo que represente mejor el problema. Para finalizar la se presenta las formas de encarar este problema, sección 2.1 y el modelo de supermaximales dado en [**12**], sección 2.2, que será la base de nuestro trabajo.

Propuesta: Describe la teoría para demostrar que el algoritmo es correcto. Primero plantea el problema formalmente, sección 3.1; luego propone un álgebra de repeticiones para describir que es una repetición supermaximal en éste problema, sección 3.2. Se asocia el comportamiento de un Árbol de Sufijos de Supermaximales con el Álgebra de Repeticiones para demostrar

que es una estructura válida para representar nuestro problema, sección 3.3, y se determina los algoritmos para manejar la estructura, sección 3.4.

Resultados y conclusiones: En este último capítulo se presentan detalles que se tuvieron en cuenta en la implementación del AS^2, sección 4.1. Se describe en la sección 4.2 diferentes resultados obtenidos de aplicar el algoritmos sobre genomas que han y están siendo secuenciando con técnicas aleatorias. Por último, puede encontrarse las conclusiones de este trabajo en la sección. 4.3.

1.2. Otros Fundamentos.

Se ha comparado al Proyecto Genoma Humano con los proyectos más ambiciosos de la humanidad: la llegada a la luna o la construcción de las pirámides. La cantidad de recursos que se necesita para poder alcanzar el objetivo es abrumador, y a pesar de ello el mundo científico esta convencido de su utilidad médica y tecnológica, mas allá del carácter filosófico que pueda tener. Tanto es así que el Proyecto Genoma Humano, completado cinco años antes de lo previsto **[25]**, ahora se extiende a miles de proyectos genomas que van desde virus hasta mamíferos, incluyendo especies ya extinguidas **[24]**.

Pero aun así existen problemas muy difíciles de abordar, incluso teniendo a disposición tantos recursos. Dentro de éstos están los inducidos por las *repeticiones*. Tal es así que un genoma tan importante para Sudamérica como el del *Trypanosoma cruzi* se ha interrumpido por la gran cantidad y diversidad de estos elementos**[27, 7]**. Esto da pie a la necesidad de su estudio dando un nuevo punto de vista. Esta tesis aborda el problema de buscar estos elementos en un conjunto de fragmentos, y para ello plantea un nuevo modelo de repeticiones. Esto no quiere decir que es la única forma de buscar repeticiones que se conoce, sino que intenta establecer su búsqueda en un campo de batalla en que no se ha intentado antes: durante el proceso de secuenciación. Los algoritmos conocidos y las técnicas biológicas intentan encontrar estos elementos en el genoma completo o directamente sobre la molécula de ADN, respectivamente. Pero esto produce contradicciones insalvables: ¿Cómo buscar repeticiones en el genoma completo si estas mismas repeticiones no nos permite completarlo? ¿Cómo podemos leer una secuencia repetida de un ADN cuando ésta es más larga de lo que nos permite las técnicas de secuenciado?

Buscar las repeticiones durante el proceso de una secuenciación aleatoria permite no caer nuevamente en las contradicciones anteriores. La aleatoriedad nos asegura la completa secuenciación de genoma en forma de fragmentos, y por lo tanto no

habrá que esperar al ensamblado para conocer estos elementos. Pero igual existe una pregunta que no parece tener una solución obvia. ¿Se puede ensamblar una secuencia repetida a partir de sus fragmentos? No es objetivo de esta tesis responder esta pregunta pero no se puede hablar de búsqueda de repeticiones en fragmentos sin haberla planteado. Esta tesis confía en la afirmación de la respuesta, con lo que se espera que un trabajo a futuro pueda solucionar este problema.

Como se verá más adelante se podrán encontrar todas las repeticiones maximales en un conjunto de fragmentos obtenidos sin un orden específico. Hay que notar que aunque no se tiene la respuesta al problema de secuenciar un genoma con grandes cantidades de repeticiones, este trabajo propone un nuevo camino que puede ser el primer paso a la solución tan esperada.

1.3. Proyectos Genomas

Los Proyectos Genoma tienen como objetivo obtener información genómica de una especie para su eventual estudio. La misma esta almacenada en el *ADN*, macromoléculas capaces de guardar toda la información evolutiva, estructural y funcional de todas las células que componen a un individuo. Las mismas están estructuradas por dos cadenas de nucleótidos, una inversa-complemento de la otra, formando una doble hélice. Para este trabajo el concepto básico que usaremos es que el ADN puede leerse por fragmentos gracias al proceso denominado *secuenciación*. **[34]**

Simplificando de manera informativa, completar un proyecto genoma necesita de una continua realización de tareas de secuenciación, ensamblado y anotación. La *secuenciación* es un procedimiento analítico para obtener el orden de los nucleótidos de un ADN[2] **[6]**, o de un punto de vista computacional es la tarea de traducir el DNA en información digital, o computacionalmente tratable. La información obtenida es una secuencia de símbolos (A, C, G, T); cada uno representa a uno de los cuatro *nucleótidos*: adenina, citosina, guanina y timina -también conocidos como *bases*-. Esta tarea está limitada técnicamente a la obtención de secuencias de 400 a 700 símbolos de largo. Comparado con el genoma de una especie de $5 \cdot 10^{11}$ *pares de bases* (pb) de largo, es casi insignificante lo que se puede realizar en una sola lectura, apenas un $1,4 \cdot 10^{-7}$ % del mismo. Por eso existe la tarea de *ensamblado*; que se encarga de *clasificar*, *ordenar* y *concatenar* los fragmentos con el objetivo de obtener la secuencia completa. La *notación*, en cambio, intenta dar *sintaxis* y *semántica* al genoma **[26]**.

[2]Este procedimiento no se limita únicamente a cadenas de nucleótidos, sino también a cadenas de aminoácidos. Pero nuestro objetivo es trabajar con nucleótidos, ya que las repeticiones en cadenas de aminoácidos tienen implicancias completamente diferentes que están fuera del alcance de la tesis.

Las secuencias de 400 a 700 pb, nombradas anteriormente, se las conoce como fragmentos ya que son una parte del DNA y se obtiene por medio de métodos de *fragmentación* y *clonación*.

Todas las tareas están estrechamente relacionadas, por ende un problema con cualquiera de éstas puede inducir al fracaso del proyecto. ¿Pero qué problemas podemos encontrar durante todo este proceso?

***Secuenciamiento*:** El principal problema del secuenciamiento es la estructura natural del DNA. Existen diferentes estructuras que promueven su plegamiento obstruyendo la lectura. Este problema se resuelve usando diferentes *geles*[3] como soporte del secuenciado. Pero no existe una única solución, con lo que puede ocurrir que para cada fragmento sea necesario cambiar de gel. Estas regiones se las reconoce por tener alto contenido de G-C (Guanina y Citosina), ser palíndromos[4] o homopoliméricas[5]. Un segundo problema, pero de menor medida, es secuenciar regiones no deseadas. La secuenciación involucra un proceso previo de preparación de los fragmentos. Estos se los selecciona de una *biblioteca*, conjunto de fragmentos seleccionados para el secuenciamiento. Cada fragmento esta contenido en un *vector* o *BAC*: soporte para realizar el proceso de clonado necesario para la secuenciación. El problema es que el vector también es DNA, por lo tanto se puede leer tanto información del soporte como del fragmento. Este problema se resuelve filtrando las secuencias correspondientes al vector. Luego podemos encontrar los problemas clásicos de error por limitaciones técnicas de los métodos usados.**[34, 31]**

***Ensamblado*:** A diferencia de la secuenciación, los problemas no son bioquímicos sino lógicos. A partir de una base de datos depurada[6] donde ya se han resuelto los problemas de la secuenciación se deben obtener la secuencia

[3]Los geles son materiales bioquímicos en donde se sumerge el DNA para ser secuenciado.

[4]Según la Real Academia Española: palabra o frase que se lee igual de izquierda a derecha, que de derecha a izquierda; p. ej., *anilina*, *dábale arroz a la zorra el abad*.
En bioquímica: una secuencia de DNA que es igual a su inversa-complemento, por ejemplo: *ACTGCAGT*.

[5]Secuencias donde predomina un nucleótido.

[6]Bases de datos de secuencias que fueron tratadas para eliminar regiones no deseadas: baja calidad de nucleótidos, vectores y errores.
Se la puede encontrar en la bibliografía como "base de datos curadas", ya se lo ha traducido literalmente del ingles "curated databases". La traducción elegida, *base de datos depurada*, expresa mejor el concepto.

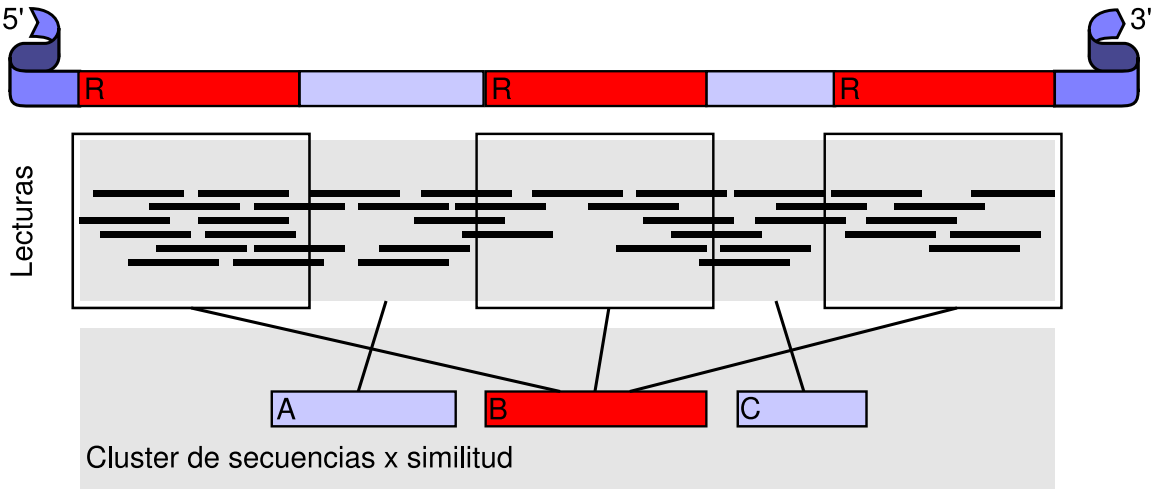

FIGURA 1.3.1. Esquema de generación de un cluster falso. La figura representa cinco regiones, tres de ellas, R, son copias de una misma repetición. Relacionando las lecturas por similitud, se obtienen tres clusters. El cluster B unifica las tres regiones correspondientes a la repetición R, por lo tanto esta configuración de contigs es falsa.

completa del genoma. Para ellos se va construyendo fragmentos más grandes, combinando tareas de clasificación y ensamblado. La *clasificación* tiene como objetivo identificar las secuencias que corresponden a una misma región para obtener una *secuencia consenso* que la represente.
El *ensamblado* tiene como objetivo poder concatenar estas secuencias consenso; la secuencia obtenida se la conoce como *contig*.
Cuando trabajamos con repeticiones con largos mayores al límite de los que nos proporcionan los secuenciadores (700 pb) o repeticiones en los extremos de los consensos, nos encontramos con falsas clasificaciones y ambigüedades en el armado de contigs respectivamente. Un cluster es falso si se toma secuencias que corresponden a diferentes regiones del genoma como de una misma región (Ver figura 1.3.1). Y las ambigüedades se produce por no determinar de forma univoca como ordenar los contigs que contienen una repetición en sus extremos (Ver figura 1.3.2).**[26, 31, 29]**

***Anotación*:** Existen diversas complicaciones a la hora de plasmar las conclusiones o investigaciones en una región exacta del genoma y todo se debe a

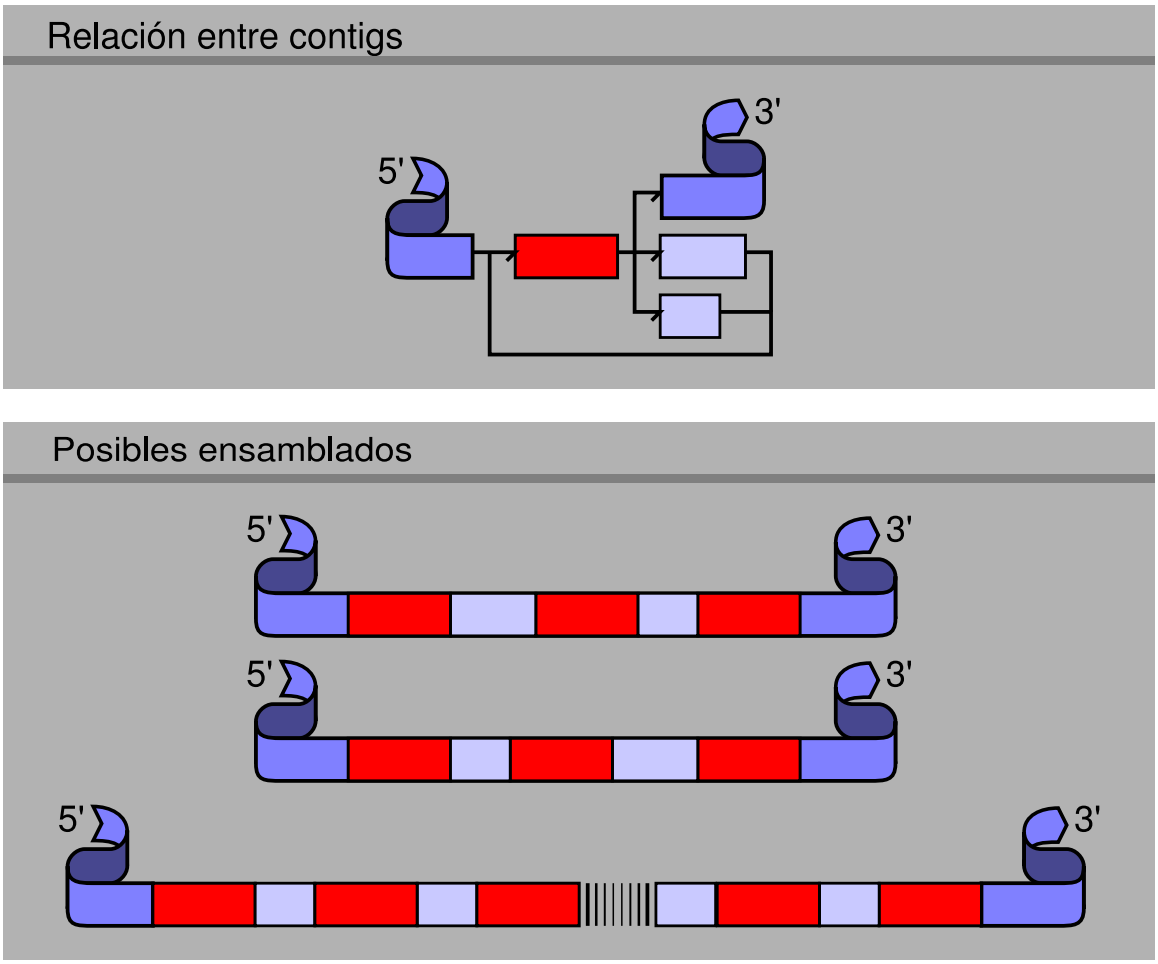

FIGURA 1.3.2. Esquema de ambigüedad. A partir de la figura 1.3.1 se tienen cinco clusters. Si se intenta ordenas estos clusters se tiene infinitas soluciones posibles si desconocemos que los clusters de azul claro no son repeticiones. En el caso que sabemos que no lo son, existen solo dos formas de ordenar los clusters.

la complejidad del modelo tratado. Por ejemplo, actualmente las bases de datos genómicas almacenan miles de *proteínas hipotéticas* provocando entre los investigadores diferentes interpretaciones **[32, 8]**. La imposibilidad de clasificar e identificar cada elemento del genoma de manera unívoca a enviciado

a publicar resultados obtenidos computacionalmente, muchas de ellas con técnicas heurísticas, como verdades absolutas. Y esto ocurre por la desinformación sobre estas técnicas, sumado a que todavía el genoma es un laberinto oscuro, declarar a las herramientas computacionales como funcionalmente válidas es dejar pistas falsas en la interpretación del genoma.**[31, 26]**

Como se puede observar, hemos hecho hincapié en las repeticiones. Esto no implica que las repeticiones son la única fuente de problemas, pero si de su gran mayoría, y puede encontrarse en cada etapa del proceso de un proyecto genoma.

1.4. Secuenciamiento Ordenado vs. Aleatorio.

Existen dos tipos básicos de *secuenciamientos* para encarar proyectos genoma: *ordenado* y *aleatorio*. Ambas pueden complementarse para completar un proyecto genoma ya que cada uno tiene virtudes y defectos contrapuestos.

Caminado: La primera técnica se basa en la existencia de primers. Un *primer* es una secuencia única dentro de una región o fragmento del genoma y sirve para marcar el comienzo del secuenciamiento. El proceso de *caminado* conlleva una continua búsqueda de primers y secuenciación de fragmentos adyacentes (Ver figura 1.4.1. Aunque es teóricamente sencilla la tarea es ardua y consume una cantidad muy alta de recursos, además de estar condenada a tiempos imposibles cuando los genomas son medianos o grandes, ya que cada paso puede durar días. Se usa en regiones pequeñas y para resolver un ensamblado**[28, 31]**

Shotgun: El *secuenciamiento aleatorio*, o *shotgun*, empieza fragmentando una parte, o todo el genoma en un conjunto de piezas al azar de longitudes conocidas. Esto nos permite realizar secuenciamiento en paralelo con lo que se acelera drásticamente el procesamiento (Ver figura 1.4.2). Aún así el ensamblado termina siendo un proceso más complejo, aunque automatizable. Para resolver los problema de espacios no secuenciados (*GAPs*) o *subsecuenciados*, ver figura 1.4.3, se asegura que la cantidad de lecturas sobre la región a secuenciar sean suficientes gracias a los estudios estadísticos de Landerman. Si esta estadística no satisface, que es posible, se completa el secuenciamiento con la técnica de caminado. **[28, 31]**.

El número en promedio en que un nucleótido esta representado por una base de alta calidad en una colección de secuencias obtenidas aleatoriamente se lo conoce como *cobertura*. En otras palabras, la cobertura es la relación entre la cantidad de

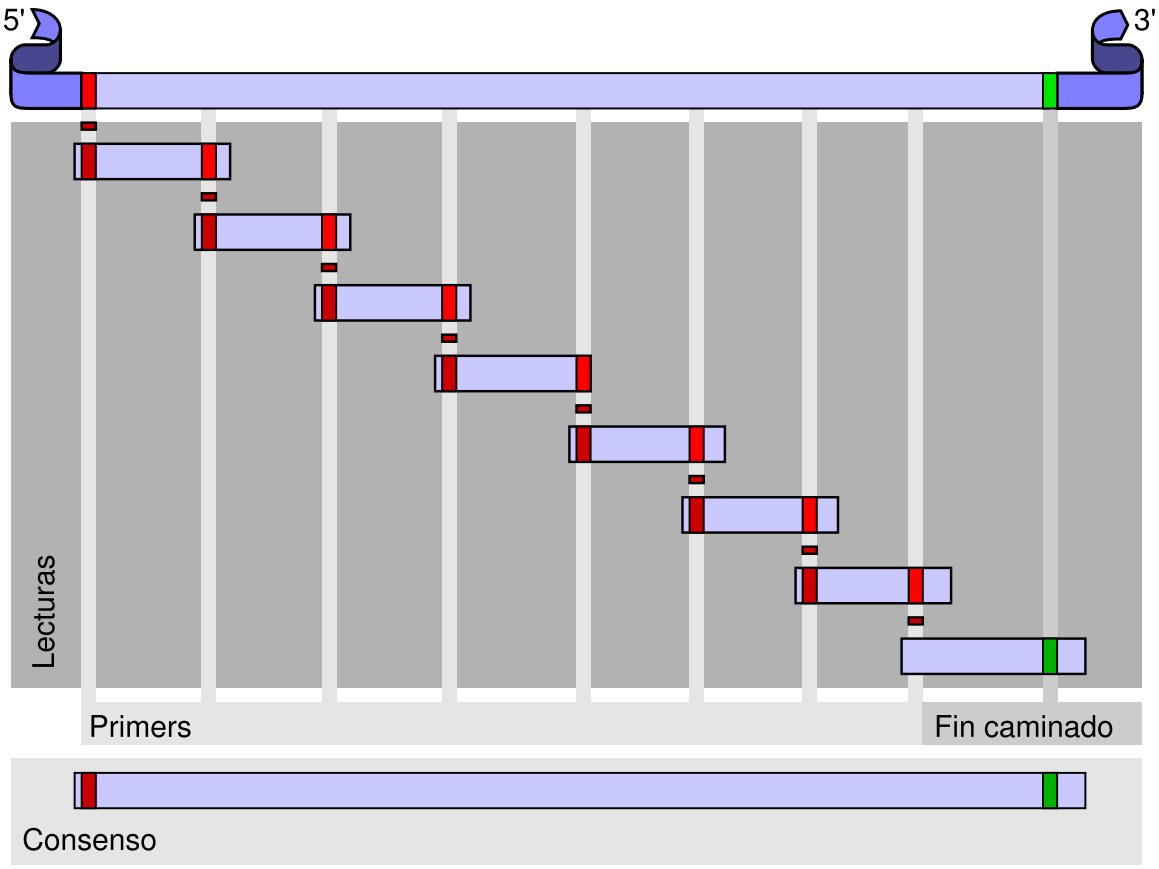

FIGURA 1.4.1. Secuenciamiento ordenado. El proceso de caminado se realiza a partir de un *primer* inicial que indica el principio de la región del genoma a secuenciar. Luego se realizar lecturas sucesivas a partir de los nuevos primers que se valla "recolectando" en el camino.

nucleótidos que debemos leer y la cantidad de nucleótidos que compone nuestro fragmento a secuenciar (Ver figura 1.4.4). Una simple fórmula descripta en **[17]**, puede estimar la cobertura necesaria para secuenciar aleatoriamente una parte del genoma para prevenir los problemas del ensamblado. El cuadro 1 muestra valores usados para aproximar la cobertura. Por ejemplo si se obtiene 100.000 *pares de bases* (pb) de un clon de 100 Kpb, se espera haber secuenciado el 63 % del clon, ya que 100.000 es una

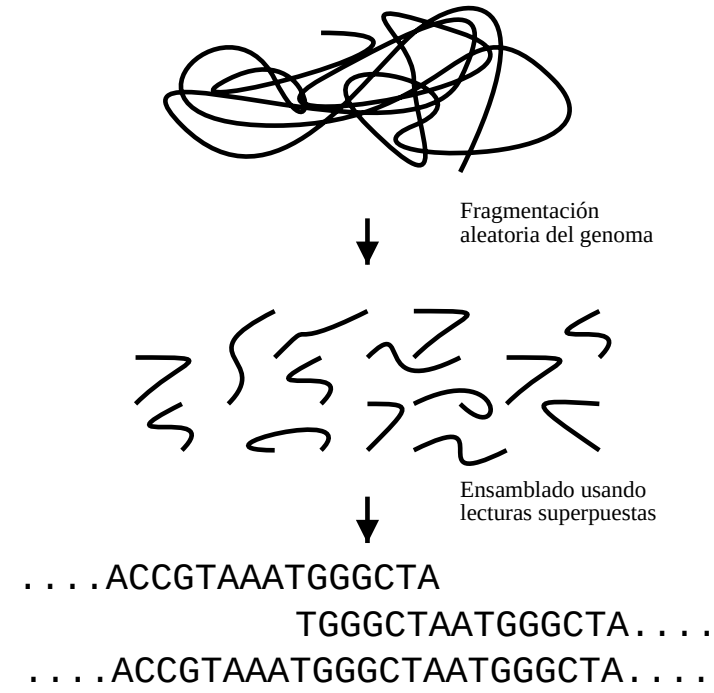

FIGURA 1.4.2. Secuenciamiento aleatorio. Los procesos de secuenciamiento *shotgun* fragmentan el genoma de manera aleatoria, para luego leer todos los fragmentos para ensamblarlos. Si la técnica fragmenta todo el genoma aleatoriamente se la conoce como *Whole Genome Shotgun*, WGS. En caso de que la fragmentación sea por cromosoma se lo conoce como *Whole Chromosome Shotgun*, WCS. Otras técnicas fragmentan el cromosoma a partir de marcas, *BAC ends*, que luego se usan para ordenar los fragmentos una vez que se ensamblen a partir de un secuenciamiento aleatorio. Esta estrategia se la conoce como 'Map-as-you-go'.

ves (1x) la cantidad de pb del clon. Por lo tanto es muy común ver que la mayoria de los genomas son secuenciados bajo una cobertura de entre 8x y 10x.

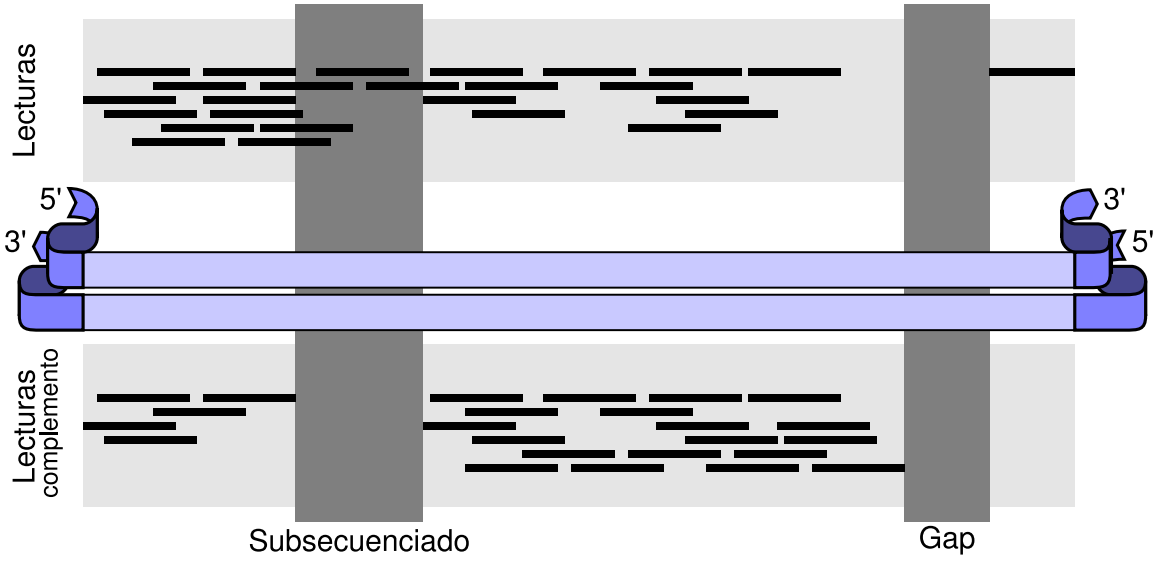

FIGURA 1.4.3. Esquema de regiones subsecuenciadas. En este gráfico se presenta una región que fue secuenciada de forma aleatoria. Los fragmentos superiores fueron secuenciados de 5á 3', mientras que los inferiores fueron secuenciados de 5á 3', su complementario. Se describe en la figura dos regiones, una que tiene un espacio no secuenciado (GAP) y otro subsecuenciado al cual falta secuenciar la hebra complementaria (Hebra simple).

1.5. Secuencias repetidas

En [**11**] se describe la organización del genoma en dos tipos de elementos, *genes de copia única* y *DNA repetitivo*. Este último a su vez organizado en diferentes subclases, ver figura 1.5.1.

Repeticiones funcionales: Algunas funciones genéticas se presentan en múltiples copias. Todas ellas pertenecen a alguna de estas clases.

Familia de genes dispersos: Muchas genes que codifican en proteínas se agrupan en familias por homología. Genes de una misma familia pueden tener leves diferencias aún cumpliendo la misma función. Aunque existen casos en que las funciones pueden diferir.

Arreglos de familia de genes en tándem: Una manera de satisfacer las necesidades de la célula de gran cantidad de productos de algunos

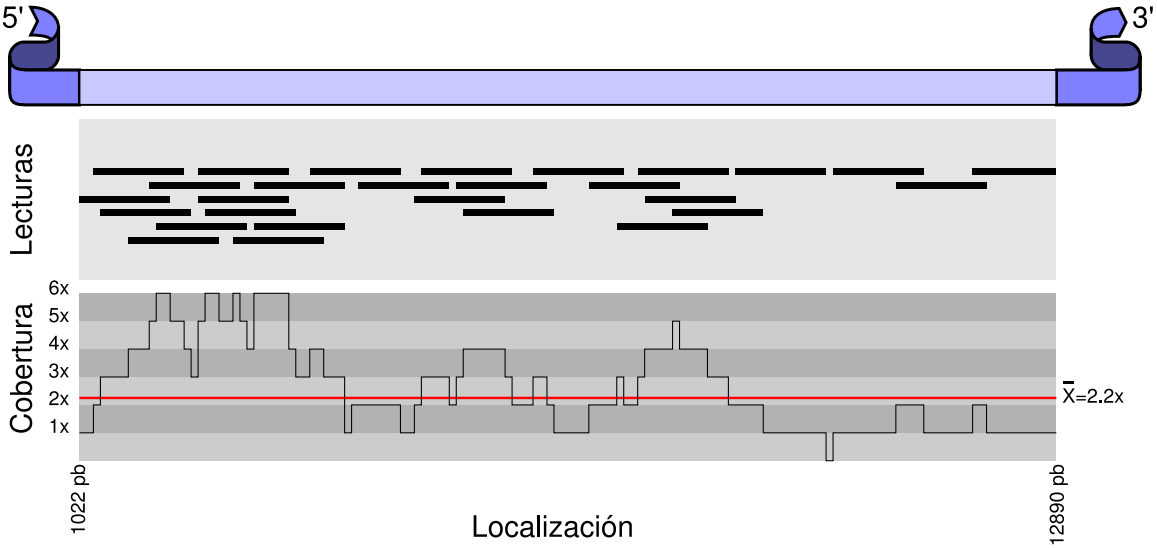

FIGURA 1.4.4. Esquema de cobertura. En la figura se representa un región del genoma que fue secuenciada aleatoriamente. El gráfico inferior describe, nucleótido a nucleótido, la cobertura correspondiente. La cobertura media es de $2{,}2x$.

genes es organizar los mismos en tándem. O sea, múltiples copias adyacentes de un mismo gen.

Secuencias que no codifican: Los telómeros son una región del genoma que contiene secuencias que no codifican, no obstante estas regiones son de vital importancia para la estabilidad del cromosoma. Estos se componen de una gran cantidad de copias de secuencias muy cortas de DNA organizadas en tándem.

Repeticiones sin función conocida: Las repeticiones que no tengan funcionamiento conocido se las organizan de la siguiente manera.

DNA altamente repetitivo del centrómero: Se puede encontrar una gran cantidad de repeticiones, organizados en tándem, que flaquean el centrómero de los cromosomas.

VNTR: Son una clase especial de repeticiones en tándem de entre 15 y 100 nucleótidos. Se encuentran dispersos por todo el genoma.

Transposones: Una gran proporción del genoma eucariota están compuestos de elementos que son capaces de propagarse. A través de un

Cobertura	Porcentaje del clon secuenciado
0,25x	22 %
0,50x	39 %
0,75x	53 %
1x	63 %
2x	88 %
3x	95 %
4x	98 %
5x	99,4 %
6x	99,75 %
7x	99,91 %
8x	99,97 %
9x	99,99 %
10x	99,995 %

CUADRO 1. Relación entre la cobertura y el porcentaje del clon secuenciado.

proceso de copia estas secuencias llamadas transposones puede reubicarse en otras regiones. Si las secuencias que se insertan en el genoma provienen de un RNA, se los conoce como retrotransposones.

A partir de esta descripción de las repeticiones podemos llegar a caracterizar atributos comunes a las repeticiones: largo, cantidad, conservación y dispersión. El *largo* y la *cantidad* son atributos triviales, y no así el resto que pasaremos a detallar.

Conservación: Las copias pueden variar levemente entre ellas. Las diferencias son las que determina el *grado de conservación*. Este atributo puede variar dentro de la secuencia de manera tal de identificar regiones con menos cambios que otras. Se puede ver esta complejidad en la estructura en familias de genes, como es el caso de VIPER (Ver figura 1.5.2) en el *T. cruzi* o la familia de genes que codifican la hemoglobina en el humano (Ver figura 1.5.3). En cambio las VNTRs son muy conservadas.

Dispersión: Indica como se distribuyen las copias en el genoma. Puede ser *local* a una región bien definida, como es el caso de las repeticiones centroméricas o teloméricas. Como también se las puede encontrar *dispersas* como los transposones.

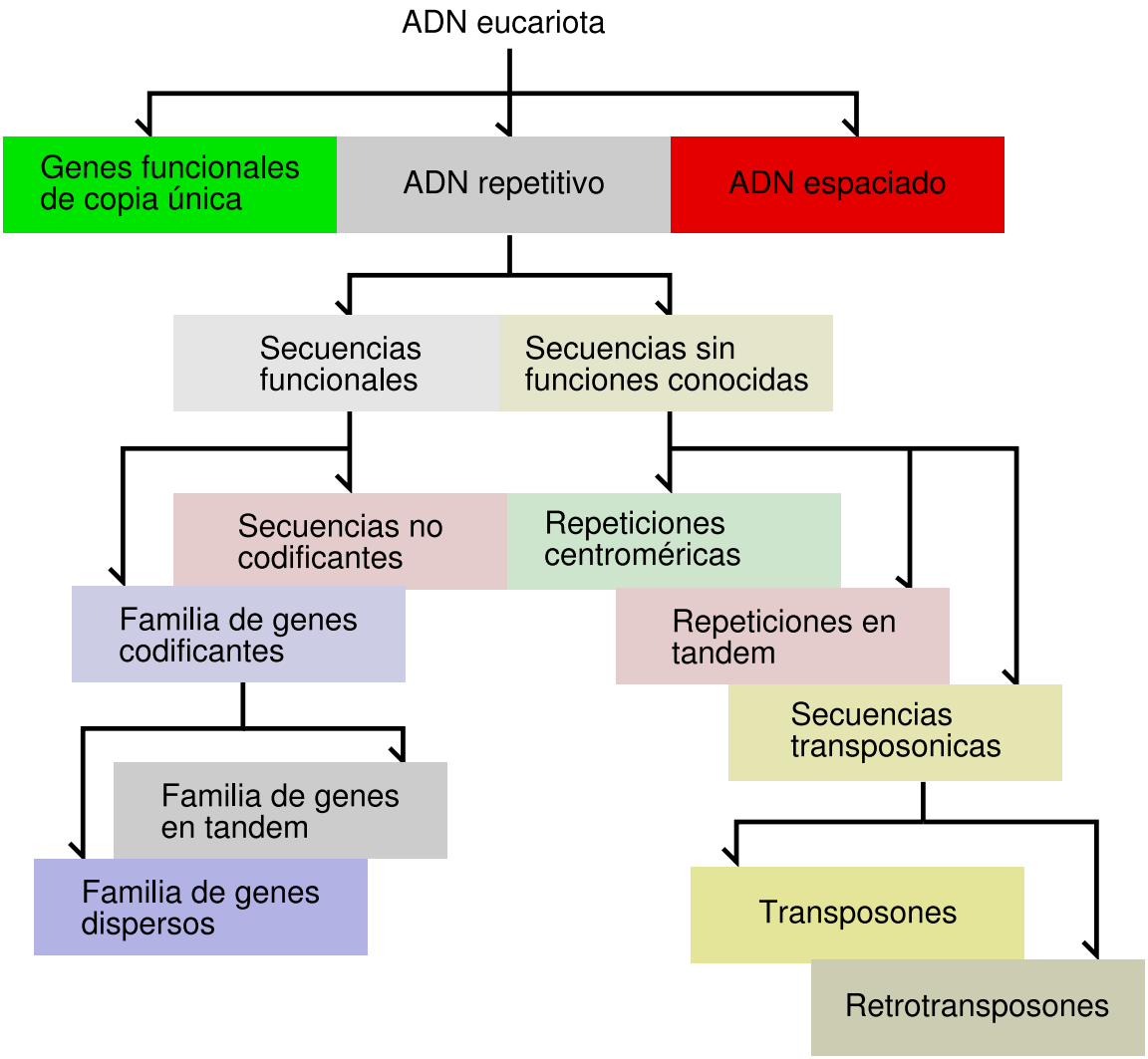

FIGURA 1.5.1. Organización de las moléculas de ADN eucariota. (ver [**11**])

1.6. Previas.

Antes de comenzar a describir más el problema, es necesario realizar varias aclaraciones sobre la notación usada en el texto. En principio los términos *símbolo*, *letra* y *carácter* se entenderán como sinónimos, refiriéndose a la entidad mínima de lo que

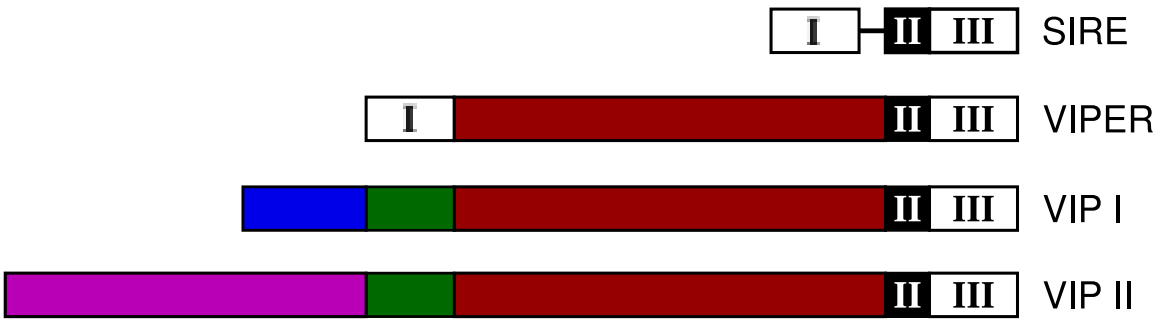

FIGURA 1.5.2. Organización de VIPER, retroelemento relacionado con retrotransposones LTR, ver **[19]**.

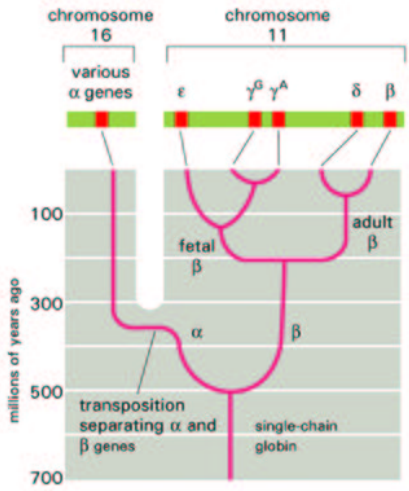

FIGURA 1.5.3. Evolución de la familia de genes de la hemoglobina en el genoma humano.

están compuesto las secuencias. Así también se verán intercalados los términos *secuencia, palabra* y *cadena* para referencia a una sucesión ordenada de esta entidad mínima. En cambio, *fragmento* y *subsecuencia* son conceptos diferentes. Un fragmento, como se ha dicho, es una secuencia de DNA o proteína que ha sido obtenido por métodos de clonación o cortando una secuencia mayor. A diferencia de fragmento,

subsecuencia es una referencia a una región de una secuencia. La misma no tiene sentido si no hay una secuencia que la contenga.

Desde el punto de vista biológico un elemento repetido es una clase de fragmentos homólogos dentro de un mismo genoma. Pero la definición de *repetición* en una *cadena de símbolos* no es tan amplia ya que se toma el concepto de igualdad estructural y no de homología para identificarlas. En consecuencia se usa el concepto de similaridad, que aunque es una aproximación a la homología da excelentes resultados.

CAPíTULO 2

Estado del arte

Lo que no se ha encontrado durante el proceso de investigación de esta tesis es información o artículos sobre la obtención o estadística de estos elementos durante el proceso de secuenciamiento. La totalidad de las referencias involucran el genoma completo **[31]**, por lo que cualquier estudio es válido si se completa el secuenciamiento. Es por eso que la totalidad de las referencias de algoritmos encontradas tienen como precondición una única secuencia de donde se busca las repeticiones.

2.1. Trabajos relacionados.

Además de los artículos relacionados con la búsqueda exacta de repeticiones en una cadena de caracteres, existe un número considerable de escritos que se especializan en la detección de repeticiones degeneradas [1]. En general los métodos se dividen en dos grupos.**[15]**

Métodos exactos: Se basan en la formalización de un modelo de repetición, y luego localizan todas las regiones en una secuencia que cumplen con la definición. **[9, 18, 3, 14, 33, 4, 16, 21]**

Métodos heurísticos: No pueden garantizar encontrar todas las repeticiones bajo cualquier modelo específico.**[5, 1, 30, 36, 4, 2]**

Tampoco se ha encontrado referencias al problema de encontrar repeticiones en una colección de fragmentos de secuencias genómicas. No así, en cambio, sobre optimización de búsquedas de sufijos y prefijos de secuencias en base de datos en general**[13]**, que en su objetivo de optimizar la consulta utilizan índices de subcadenas en común siendo referencias importantes para resolver el problema. Entonces se propone dividir, nuevamente, este tipos de problemas de búsqueda de repeticiones en:

Única secuencia: La mayoría de los trabajan bajo este paradigma ya que se esfuerzan en buscar repeticiones en una región.

[1] En la literatura sobre cadena de caracteres se conoce a las repeticiones degeneradas como repeticiones aproximadas o inexactas.

Conjunto de secuencias: Existen trabajos relacionados como el caso de los árboles de sufijos generalizados (ver sección 2.2.3), pero el objetivo en general es buscar palabras en textos, pero no se ha encontrado un uso genómico para este algoritmo.

Aún así, para describir mejor el objetivo de esta tesis hay que incluir una nueva clasificación para búsquedas de repeticiones en un conjunto de secuencias.

Secuencias independientes: Las secuencias del conjunto de entrada no están superpuestas. Todos los algoritmos existentes trabajan bajo esta suposición.

Secuencias dependiente: Las secuencias son subsecuencias de una misma secuencia y están superpuestas. Llamaremos *cobertura* a la cantidad de veces que una posición de la secuencia mayor se encuentra representada en las subsecuencias. Por ejemplo, la mayor cobertura del conjunto $abcc, yuab, ccak, uab$ de la secuencia $\alpha = yuabccak$ es de $3x$, ya que ab se encuentra 3 veces en α. La menor cobertura es de $1x$, ya que k solo se encuentra una única vez en el conjunto. La cobertura promedio se calcula símbolo a símbolo, por lo que en promedio α tiene una cobertura de $1{,}875x$. No se ha encontrado trabajos previos.

Bajo esta clasificación, esta tesis se concentra en la *búsqueda de repeticiones exactas en un conjunto de secuencias dependientes*. Para justificar la elección de repeticiones exactas debemos responder la siguiente pregunta: ¿Qué tan similar es una secuencia a otra? Existen varios modelos de distancia que tienen en cuenta mutaciones, equivalencia entre nucleótidos, llegando a ser probabilísticamente complejos. Aún así, tomando cualquiera de estas forma para medir la similitud, no se puede asegurar que dos secuencias similares correspondan a dos fragmentos homólogos. Basándose en éste hecho se puede suponer que la igualdad entre secuencias es una buena aproximación al concepto de homología, por ello en el resto de la tesis solo se hablará de igualdad entre secuencias para identificar elementos repetidos. Esta afirmación no quita valor a las normas de similitud existentes, solo simplifica el problema que interesa tratar para poder, a futuro, plantear un modelo más complejo.

2.2. Modelo de Gusfield de repeticiones

Un problema a la hora de identificar secuencias repetidas es determinar cuando una secuencias es repetida, o mejor dicho, cuando una repetición es importante. En principio la secuencia vacía es repetición para cualquier cadena de cualquier lenguaje, por lo tanto es una repetición trivial y no se tendrán en cuenta. Otras repeticiones

triviales son las de largo uno, los símbolos. También se incluyen todas las subsecuencias de una secuencia repetida y así se tiene una cantidad desorbitante de soluciones sin valor informativo. En pos de definirlas de manera tal que las soluciones sean significativas se ha incluido el concepto de *maximal* dando las siguientes definiciones[**12**].

DEFINICIÓN 2.2.1. Un *par maximal* (o un par de repetición maximal) en una cadena S es un par de subcadenas idénticas α y β en S tal que el carácter inmediatamente posterior (anterior) de α es diferente al carácter posterior (anterior) de β. De otra manera, si extendemos α y β en ambas direcciones dejan de ser iguales. Un par maximal esta representado por una tupla $< p_1, p_2, n\prime >$, donde p_1 y p_2 indican la posición de inicio en ambas subcadenas y $n\prime$ indica el largo. Para una cadena S definimos $R(S)$ como el conjunto de todos los pares maximales de S.

Eligiendo la secuencia $S = xabcyiiizabcqabcyrxar$, se encuentran tres ocurrencias de la secuencia abc. La primera y la segunda ocurrencia conforman el primer par maximal $< 2, 10, 3 >$. La segunda con la tercera conforman el par $< 10, 14, 3 >$. Pero no es par maximal la primera con la tercera ocurrencia. En cambio, si se toma la secuencia $abcy$ se obtiene que sus dos únicas ocurrencias son el par maximal $< 2, 14, 4 >$. Ahora, si la secuencia es $cxxaxxaxxb$ observamos un par maximal solapado correspondiente a la secuencia $xxaxx$. Notar que este caso es permitido por la definición.

Ahora, en algunos casos es útil conseguir todas los pares maximales de una secuencia, en otros casos la cantidad de soluciones puede ser inmanejable, por lo tanto sería útil definir un concepto más restringido.

DEFINICIÓN 2.2.2. Una *repetición maximal* α es una subcadena de S que corresponde a un par maximal de S. En otros términos, α es una repetición maximal en S si hay una tupla $< p_1, p_2, |\alpha| > \in R(S)$ y α aparece en las posiciones p_1 y p_2 en S. Diremos que $R\prime(S)$ es el conjunto de repeticiones maximales de S.

Del ejemplo anterior con la secuencia S donde ambas cadenas abc y $abcy$ son repeticiones maximales, se puede observar que la cantidad de pares es mayor que la cantidad de repeticiones. Esto ocurre porque cada cadena esta representado por lo menos un par maximal, entonces concluimos que $|R(S)| > |R\prime(S)|$. En general la cantidad de $|R\prime(S)|$ va a ser muy inferior que $|R(S)|$ por lo que, aunque este conjunto es más modesto, da una idea clara de la solución al problema.

Ahora, existen casos en que la definición de repetición maximal no es suficiente. Tomando una nueva secuencia $S = a\alpha bx\alpha ya\alpha b$. Aquí la subcadena α es un repetición maximal como lo es también $a\alpha b$, la cual es supersecuencia de α. Quizás no se desea

reportar secuencias α como repetición ya que existe una repetición maximal que la contiene que puede ser más informativa.

DEFINICIÓN 2.2.3. Una *repetición supermaximal* es una repetición maximal que no es subcadena de otra repetición maximal.

Para Gusfield, las repeticiones maximales y supermaximales y los pares maximales son las únicas tres maneras válidas para definir estructuras útiles de repeticiones exactas. Esta afirmación solo es válida en el modelo de buscar repeticiones en una única secuencia, a continuación se planteará los mismos conceptos para un conjunto de secuencias.

2.2.1. Modelo de Gusfield generalizado. Tomemos ahora un conjunto de secuencias S. Para cada secuencia $s \in S$ podemos usar las misma definiciones de par maximal, repetición maximal, y repetición supermaximal, pero no ocurre lo mismo para un conjunto de secuencias ya que lo que para una secuencia no es repetida para s puede que lo sea para un conjunto si es compartida por otra secuencia. Para ello generalizaremos las definiciones de par maximal, repetición maximal y supermaximal para un conjunto de secuencias. Previamente se definirá a $\alpha(\pi, \sigma)$ como la subsecuencia de α que comienza en la posición π y termina en σ. Y S_τ es la secuencia identificada con τ en el conjunto de secuencias S.

DEFINICIÓN 2.2.4. Un *par maximal* es un par de subsecuencia idénticas α y β de secuencias existentes en S las cuales dejan de ser iguales si se las extiende en ambos lados.

Un par maximal del conjunto S se representa por la tupla $< id_a, id_b, p_a, p_b, n\prime >_S$, donde id_a y id_b son la identificación de las secuencias en S, p_a y p_b son la ubicación de la repetición en la secuencia S y $n\prime$ es el largo. Para un conjunto S definimos $R(S)$ como el conjunto de todas las tuplas maximales de S.

$$\begin{aligned} R(S) &= \{< id_a, id_b, p_a, p_b, n\prime > \mid \\ (id_a &\neq id_b \vee p_a \neq p_b) \wedge \\ S_{id_a}(p_a, p_a + l) &= S_{id_b}(p_b, p_b + n\prime) \wedge \\ S_{id_a}(p_a, p_a + n\prime + 1) &\neq S_{id_b}(p_b, p_b + n\prime + 1) \wedge \\ S_{id_a}(p_a - 1, p_a + n\prime) &\neq S_{id_b}(p_b - 1, p_b + n\prime)\} \end{aligned}$$

A partir del del conjunto de secuencias $S = \{xabclop, lapabx, pabcjabp\}$ se concluye: la secuencia ab corresponde para los siguientes pares maximales $< 1, 2, 2, 4, 2 >$,

$< 1, 3, 2, 6, 2 >$, $< 2, 3, 4, 2, 2 >$, $< 2, 3, 4, 6, 2 >$ y $< 3, 3, 2, 4, 2 >$. Obsérvese que la definición nos permite realizar pares con la misma secuencia, incluyendo de esta manera la definición de par maximal para una única secuencia. Para el caso de la secuencia *abc* se tiene únicamente el par $< 1, 3, 2, 2, 3 >$.

DEFINICIÓN 2.2.5. Una secuencia α es *repetición maximal* de un conjunto S si existe un par maximal que corresponda a α. En si, α es una repetición maximal en S si hay una tupla $\langle i_a, i_b, p_a, p_b, |\alpha| \rangle \in R(S)$ y α aparece en las posiciones p_a en la secuencia i_a y p_b en la secuencia i_b del conjunto S. Diremos que $R\prime(S)$ es el conjunto de repeticiones maximales de S.

$$R\prime(S) = \{\alpha \in \Delta^* | (< i_a, i_b, p_a, p_b, l > \in R(S))(S_{i_b}\,(p_b, p_b + l) = \alpha)\}$$

Entonces, tomando el mismo ejemplo anterior, las secuencias *ab* y *abc* pertenecen a $R'(S)$. Nuevamente se concluye que el conjunto $R\prime(S)$ es inferior que $R(S)$, por corresponder a cada secuencia en $R'(S)$ por lo menos una tupla en $R(S)$. De la misma manera vamos reduciendo la respuesta, pero todavía se puede tener una solución demasiado abultada para tratarla fácilmente. La próxima definición limita más la respuesta, de igual manera en que lo hacía el modelo de Gusfield.

DEFINICIÓN 2.2.6. Una repetición *supermaximal* es una repetición maximal que no es subcadena de otra repetición maximal. Llamaremos $M(S)$ al conjunto de las repeticiones supermaximales.

$$M(S) = \left\{ \alpha \in R'(S) \,\middle|\, (\exists \beta, \gamma \in \Delta^*) \left(\beta\alpha\gamma \neq \alpha \wedge \beta\alpha\gamma \in R\acute{}(S) \right) \right\}$$

En este caso, podemos ver que *abc* es repetición supermaximal y no así *ab*. Así se obtiene un conjunto más pequeño, con lo que se espera que sea mucho más tratable.

2.2.2. Árboles de sufijos para la búsqueda de repeticiones en una secuencia. Los árboles de sufijos**[37, 22, 35]** y arreglo de sufijos**[20]** son estructuras de datos que facilitan la búsqueda de cadenas. Gusfield realiza una detallada discusión sobre estos elementos en **[12]**. Como el arreglo de sufijos es una interpretación del árbol de sufijos sobre un arreglo, solo se discutirá únicamente sobre los árboles de sufijos.

En principio, estos árboles de sufijos tienen una gran potencia ya que permiten diferentes tipos de consultas sobre los elementos que se encuentran almacenados en la estructura. La necesidad de explicar esta estructura es que es la base del algoritmo propuesto para resolver el problema. En si, Gusfield propone un algoritmo sobre la búsqueda de repeticiones sobre un árbol de sufijos dando las definiciones previamente

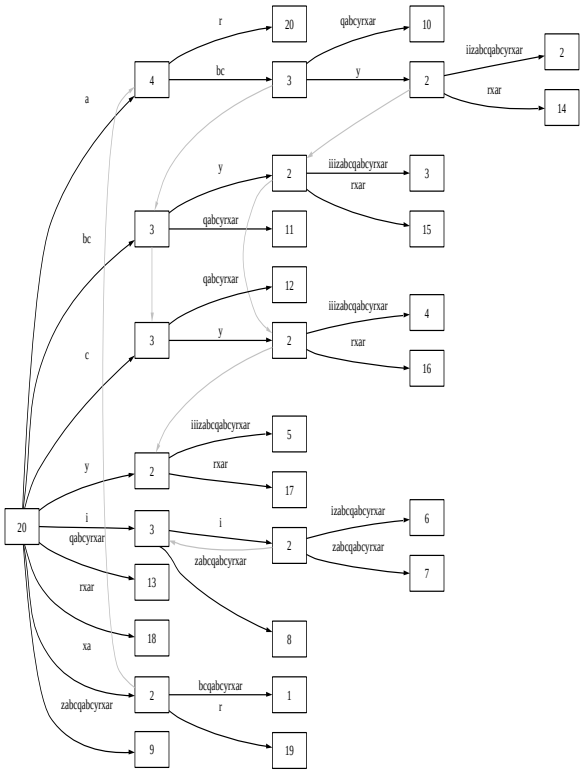

FIGURA 2.2.1. Árbol de sufijos para "xabcyiiizabcqabcyrxar".

explicadas, pero en esta nueva aproximación no se desarrolla un árbol de sufijo de una secuencia sino de las secuencias supermaximales de un conjunto de secuencias.

Los árboles de sufijos de una secuencia α son árboles, parecidos a los Trie[**10**] o PATRICIA[**23**], cuyas ramas corresponden a un sufijo[2] de α. Por ejemplo, sea la secuencia $\alpha = xabcyiiizabcqabcyrxar$. En la figura 2.2.1 vemos el árbol de sufijo para α. Si recorremos el árbol desde la raíz, en profundidad, vamos a recorrer las ramas que corresponden a los sufijos de α.

[2]Si $\alpha = a_1a_2a_3 \ldots a_{n-2}a_{n-1}a_n$, entonces $\alpha_i = a_ia_{i+1}a_{i+2} \ldots a_{n-2}a_{n-1}a_n$ es el sufijo de α en la posición i.

Por ejemplo: para $\alpha = mississippi$, tenemos los siguientes sufijos: $\alpha_1 = mississippi = \alpha$, $\alpha_2 = ississippi$, $\alpha_3 = ssissippi$, $\alpha_4 = sissippi$, $\alpha_5 = issippi$, $\alpha_6 = ssippi$, $\alpha_7 = sippi$, $\alpha_8 = ippi$, $\alpha_9 = ppi$, $\alpha_{10} = pi$, $\alpha_{11} = i$, $\alpha_{11} = i$, $\alpha_{12} = \lambda$.

Un árbol de sufijos puede ser usado para buscar una subcadena de largo m en tiempo $O(m)$. Como existen $n(n+1)/2$ subcadenas en una secuencia α de largo n no queda claro como es posible construir el árbol en un tiempo $O(n)$. Agregando solo un carácter a la secuencia α, se tienen $n+1$ nuevas secuencias, pero ellas no son independientes. Weiner (1973) dio el primer algoritmo y Mc. Creight (1976) dio una mejor aproximación para la construcción de árboles de sufijos cuando se procesa a α de derecha a izquierda. Solo más tarde Ukkonen (1992, 1995) dio un algoritmo linean procesando la secuencia de izquierda a derecha.

2.2.3. Árbol de sufijos generalizado. El árbol de sufijos que es capaz de almacenar una colección de palabras se lo conoce como *ASG*. Tiene las mismas propiedades que el árbol de sufijo para una sola secuencia, pero se diferencian por la etiqueta usada en cada uno de los ejes del árbol. Mientras que el primero almacena una colección de referencias a la secuencia y subsecuencia relacionada con el eje del árbol, el segundo solo almacena colección de subsecuencia. Como se ve, el árbol común es un árbol generalizado con una única referencia a la misma secuencia.

Gusfield propone en **[12]** dos algoritmos de orden lineal que encuentran todas repeticiones maximales y supermaximales. Este modelo conlleva tener un árbol de sufijos armado en donde se realiza la búsqueda. Luego proponen una compactación del árbol dejando solo las ramas donde intervienen repeticiones. Como podremos observar en el capítulo 3 vamos a intercalar estas nociones con ventajas relacionadas a la técnica de secuenciamiento aleatoria.

2.3. Repeticiones en el secuenciamiento.

Como se ha dicho, en un modelo aleatorio, las secuencias se obtienen de regiones aleatorias del genoma. Durante este proceso las secuencias se van almacenando en una base de datos. Esta base de datos se las conoce como base de datos de primera pasada, ya que no han sido clasificadas[3] ni filtradas[4]. Una vez revisadas, las secuencias pasan a una base de datos depurada. Este proceso termina cuando se obtenga la cobertura predeterminada 1.4.

[3]Las secuencias se clasifican por región a las que pertenecen, técnicas usadas para la clasificación y otros datos según corresponda a la especie estudiada y la técnica usada.

[4]Las secuencias se filtran de repeticiones conocidas y de basura que correspondan al vector usado para secuenciar.

¿Qué ocurre con las repeticiones durante todo este proceso? Si la repetición es conocida es eliminada para que no provoque ambigüedades en el proceso de ensamblado. En el caso contrario se verá que existe grandes ambigüedades y falsos clusters en el ensamblado. Lo ideal es no llegar al ensamblado con repeticiones.

Existe un problema innato que hace no fácil el reconocimiento en este modelo. Una cobertura de 10x implica que necesitamos haber leído cada nucleótido en promedio 10 veces. Entonces podemos esperar que una región va a tener por lo menos 10 lecturas. Es una conclusión directa suponer que si existe una repetición α, por haber sido leída 10 veces al final del procedimiento, debe encontrarse en el conjunto de las secuencias obtenidas más que esta cantidad. Si existen dos copias de una repetición, entonces deben existir c lecturas de cada una, donde c es la cobertura dada para el genoma G. Por lo tanto, podemos asegurar que si una secuencia se repite más de $2\dot{c}$ veces es una repetición en G.

Problema: Se busca encontrar repeticiones supermaximales en una colección de secuencias independientes bajo una cobertura de $2c$, donde c es la cobertura con la que se ha secuenciado la región del genoma.

CAPíTULO 3

Repeticiones en un conjunto de secuencias dependientes

3.1. Repeticiones en un modelo de secuenciamiento aleatorio

Como hemos planteado, nuestra preocupación es obtener secuencias repetidas en técnicas de secuenciamiento shotgun, y de la misma ha surgido las siguiente incógnita:

Si estamos obteniendo secuencias aleatoriamente, podemos concluir que podemos secuenciar varias veces la misma región, es más, se espera tener secuenciado al menos cx todos las bases del genoma al termino del secuenciamiento 1.4. A la vez las secuencias repetidas tienen más de una copia dentro del genoma. Entonces ¿Cómo saber si una secuencia es repetida o es un fragmento varias veces secuenciado? ¿Qué ocurre con las repeticiones cuando tenemos en cuenta que estamos en un proceso donde se va obteniendo nuevas secuencias durante todo el proceso?

Una secuencia esta repetida si se puede reconocer entre los fragmentos más de $2c$ copias de una repetición. Se puede ser más específico y decir que, si la cantidad de copias encontradas es de $2n$, entonces deben existir al menos $\frac{n}{c}$ copias en el genoma. Esta aproximación ingenua asegura detectar todas las repeticiones al final del secuenciamiento, ya que si se encuentra una repetición en un fragmento, este no salta a la vista hasta tener la cobertura cx en el mismo.

3.2. Álgebra de repeticiones.

Para describir y demostrar la corrección del algoritmo que presentaremos debemos introducir nuevas nociones sobre repeticiones que nos serán muy útiles. En su conjunto planteará una sencilla y nueva álgebra basada en expresiones regulares y en conjuntos finitos, para luego deducir propiedades sobre repeticiones.

Partiendo del modelo propuesto por Gusfield junto con la generalización propuesta para buscar repeticiones en un conjunto de secuencias [2.2.1], se plantean las siguientes definiciones sobre el conjunto de secuencias S correspondientes a un alfabeto , donde cada secuencia se identifica unívocamente por un número entero τ, y S_τ es la secuencia. Siendo S_{10} la secuencia identificada con el número 10. Una subsecuencia de S_i que

comience en la posición π y termine en σ la indicaremos como $S_i(\pi, \sigma)$. Por ejemplo, si la secuencia $axfggh$ esta identificada con el número 10, entonces $S_{10}(2, 5) = xfg$. Si π es mayor o igual a σ, $S_i(\pi, \sigma)$ da la secuencia vacía; de igual manera si π o σ están fuera del rango de la secuencia S_i.

DEFINICIÓN 3.2.1. La tupla de subsecuencias $\langle i, \pi, \sigma \rangle_S$ identifica a la subsecuencia de la secuencia i, que pertenece al conjunto S, que comienza en la posición π y termina en la posición σ.

Una *consulta* sobre una colección de secuencias S es un conjunto Q_S de tuplas de subsecuencias. Si para toda tupla de la consulta Q_S existe una secuencias α que le corresponde la tuplas $\langle i, \pi, \sigma \rangle_S$ tal que $S_i(\pi, \sigma) = \alpha$ diremos que la *consulta es válida* para el conjunto S. Para extender nuestra definición de consulta diremos que $Q_S(\alpha)$ es el subconjunto de tuplas $\langle i, \pi, \sigma \rangle_S$ de Q_S que corresponde a $S_i(\pi, \sigma) = \alpha$. El largo de la cadena α la notaremos como $|\alpha|$.

Nuestro objetivo es conseguir un Q_S que contenga solo las secuencias supermaximales de S. En otras palabras pedimos que para cada tupla $\langle i, \pi, \sigma \rangle \in Q_S(\alpha)$, consulta válida de S, exista una tupla maximal $< j, j', p, p', |\alpha| > \in R\prime(\alpha)$ tal que $j = i$ y $p = \pi$, ó $j' = i$ y $p' = \pi$.

Toda esta teoría supone que este dado de antemano un alfabeto Δ al que pertenece todas las palabras del conjunto S. Para simplificar la notación, a partir de ahora supondremos estar hablando de un único conjunto S, con lo que no se pondrá los subíndices que indican el conjunto, excepto que se indique lo contrario.

DEFINICIÓN 3.2.2. Si el conjunto A tiene más de c elementos entonces $\Phi_c A = A$, sino $\Phi_c A = \emptyset$. Esta función Φ_c la conoceremos como la función *c-repetición* sobre conjuntos.

$$\begin{aligned} \Phi_c S &= S, \#S > c \\ \Phi_c S &= \emptyset, \#S \leq c \end{aligned}$$

La función Φ_c nos permitirá descartar los conjuntos de secuencias que no correspondan a repeticiones. Llamaremos a c el *grado de repetición de la consulta* cuando el conjunto sea una consulta. También simplificaremos la notación y no escribiremos el umbral c mientras no sea necesario, ya que siempre trabajaremos con un umbral fijo.

3.2.1. Álgebra de repeticiones. Para comenzar con las definiciones, primero vamos a definir el conjunto de repeticiones sobre una secuencia y sobre una expresión regular.

DEFINICIÓN 3.2.3. Un conjunto de repeticiones (α) es un conjunto de tuplas $\langle i, \pi, \sigma \rangle$ (consulta) donde $S_i(\pi, \sigma) = \alpha$ cuando existe más de una tupla, y es vació si existe una o ninguna tupla.

$$(\alpha)_S^c = \Phi_c \{ \langle i, \pi, \sigma \rangle_S | S_i(\pi, \sigma) = \alpha \} = \Phi_c Q_S(\alpha) \tag{3.2.1}$$

Como ejemplo tenemos el conjunto S de secuencias $\{tacgua, agata, gatca\}$. La consulta (ga) esta conformado por $\{\langle 2,2,3\rangle, \langle 3,1,2\rangle\}$ si tomamos las identificación de la secuencia empezando por 1. Ahora si tomamos la consulta (gu) es vacía ya que aunque existe la tupla $\langle 1,3,4\rangle$, es única por ende gu no es repetición. Una consulta importante, pero trivial, es la correspondiente a la secuencia vacía (λ). Esta va a contener un número infinito de tuplas, ya que, como hemos descripto la tuplas cuyos π y σ que estén fuera de rango de la secuencia, o π menor a σ, describen a la secuencia vacía.

Ahora daremos más potencia a este conjunto de secuencias para no solo consultar sobre una secuencia sino que también sobre expresiones regulares [1]. Este tipo de consulta da mayor potencia ya que se puede expresar un conjunto de consultas de secuencias que cumplan con la expresión.

DEFINICIÓN 3.2.4. Un conjunto de repeticiones $(/\beta/)$ es una consulta cuyas tuplas $\langle i, \pi, \sigma \rangle$ corresponde a una secuencia repetida que cumple con la expresión regular $/\beta/$ [2]

$$(/\beta/)_S^c = \{ \langle i, \pi, \sigma \rangle_S \in (S_i(\pi, \sigma))_S^c | S_i(\pi, \sigma) \in /\beta/ \} \tag{3.2.2}$$

Tomando el ejemplo anterior, la consulta $(/t.*a/)$ esta compuesta por las tuplas $\{\langle 1,1,2\rangle, \langle 2,4,5\rangle\}$, ya que la única repetición que cumple con la expresión regular es ta. Una consulta más interesante es $(/.*/)$ que contiene las tuplas $\{\langle 1,1,2\rangle, \langle 2,4,5\rangle, \langle 2,2,3\rangle, \langle 3,1,2\rangle, \langle 2,2,4\rangle, \langle 3,1,3\rangle\} \bigcup (\lambda)$ de las repeticiones ta, ga, gat y la secuencia vacío. Notar que si hubiera otra repetición en el conjunto entonces sus tuplas estarían

[1] Las expresiones regulares las vamos a expresar entre barras '/ 'para diferenciarlas de las cadenas del lenguaje.

[2] $S_i(\pi, \sigma) \in /\beta/$

en este conjunto. Otra consulta interesante es aquella que no incorpora la secuencia vacía: $(/.+/)$.

Para relacionar ambas definiciones damos el siguiente corolario:

COROLARIO 3.2.5. *Un conjunto de repeticiones sobre expresiones regulares verifica la siguiente igualdad:*

$$(/\beta/)^c_S = \bigcup_{\alpha \in /\beta/} (\alpha)^c_S \tag{3.2.3}$$

DEMOSTRACIÓN. La nueva expresión de $(/\beta/)$ es otra forma para describir la expresión [3.2.2]. □

DEFINICIÓN 3.2.6. La operación *extensión* $(\cdot)$ sobre consultas válidas da una nueva consulta válida cuyas secuencias son repetidas y extensiones de la consulta original. La extensión se realiza a la derecha si el operador se encuentra a la derecha de la notación, o a la izquierda en otro caso.

$$(\alpha)^c_S \cdot = \bigcup_{\alpha \in \Delta} (\alpha a)^c_S \tag{3.2.4}$$

$$\cdot (\alpha)^c_S = \bigcup_{\alpha \in \Delta} (a\alpha)^c_S \tag{3.2.5}$$

COROLARIO 3.2.7. *Realizar una extensión a derecha (izquierda) a una consulta de una expresión regular β equivale a una consulta de expresión regular $\beta\cdot\ (\cdot\beta)$.*

$$(/\beta/)^c_S \cdot = (/\beta\cdot/)^c_S \tag{3.2.6}$$

$$\cdot (/\beta/)^c_S = (/\cdot\beta/)^c_S \tag{3.2.7}$$

DEMOSTRACIÓN. Comenzando por la expresión definición 3.2.4, y usando el corolario 3.2.5, podemos demostrar la igualdad.

$$\begin{aligned}(/\beta/)^c_S \cdot &= \bigcup_{\alpha\in/\beta/} (\alpha)^c_S \cdot \\ &= \bigcup_{\alpha\in/\beta/} \left(\bigcup_{a\in\Delta} (\alpha a)^c_S \right) \\ &= \bigcup_{\alpha\in/\beta./} (\alpha)^c_S \\ &= (/\beta./)^c_S\end{aligned}$$

□

Gracias a estos dos operadores podemos presentar dos nuevos conjuntos que nos ayudaran con una definición alternativa de repeticiones maximales, y además de ser el pilar del algoritmo propuesto. Las siguientes definiciones son válidas tanto para secuencias como para expresiones regulares ya que se basan en los conceptos definidos previamente.

DEFINICIÓN 3.2.8. Una *consulta extensible* a la derecha (izquierda) es una consulta cuyas tuplas están extendidas a la derecha (izquierda) en la extensión de la consulta.

$$(\alpha\rangle^c_S = \Phi^c \{\langle i,\pi,\sigma\rangle_S \in (\alpha)^c_S \,|\langle i,\pi,\sigma+1\rangle_S \in (\alpha)^c_S \cdot\} \tag{3.2.8}$$

$$\langle\alpha)^c_S = \Phi^c \{\langle i,\pi,\sigma\rangle_S \in (\alpha)^c_S \,|\langle i,\pi-1,\sigma\rangle_S \in \cdot\,(\alpha)^c_S\} \tag{3.2.9}$$

Una *consulta no extensible* a la derecha (izquierda) es una consulta cuyas tuplas no están extendidas a la derecha (izquierda) en la extensión de la consulta.

$$(\alpha]^c_S = \Phi^c \{\langle i,\pi,\sigma\rangle_S \in (\alpha)^c_S \,|\langle i,\pi,\sigma+1\rangle_S \notin (\alpha)^c_S \cdot\} \tag{3.2.10}$$

$$[\alpha)^c_S = \Phi^c \{\langle i,\pi,\sigma\rangle_S \in (\alpha)^c_S \,|\langle i,\pi-1,\sigma\rangle_S \notin \cdot\,(\alpha)^c_S\} \tag{3.2.11}$$

Una consulta cuyas tuplas estén en una consulta (no) extensible a derecha y en una consulta (no) extensible a la izquierda la denominamos *consulta completamente (no) extensible*.

$$\langle\alpha\rangle^c_S = \langle\alpha)^c_S \cap (\alpha\rangle^c_S \tag{3.2.12}$$

$$[\alpha]^c_S = [\alpha)^c_S \cap (\alpha]^c_S \tag{3.2.13}$$

$$\begin{aligned}
(vaac) &= \{\langle 1,1,4\rangle, \langle 3,1,4\rangle\} \\
(vaac)\cdot &= \emptyset \\
\cdot(vaac) &= \emptyset \\
(vaac\rangle &= \emptyset \\
(vaac] &= \{\langle 1,1,4\rangle, \langle 3,1,4\rangle\} \\
\langle vaac) &= \emptyset \\
[vaac) &= \{\langle 1,1,4\rangle, \langle 3,1,4\rangle\} \\
\langle vaac\rangle &= \emptyset \\
[vaac] &= \{\langle 1,1,4\rangle, \langle 3,1,4\rangle\}
\end{aligned}$$

FIGURA 3.2.1. Cálculo de la repetición supermaximal $vaac$ para el conjunto $S = \{vaac, waabaab, vaacaaq\}$.

Por ejemplo, tomando el conjunto de secuencias

$$S = \{vaac, waabaab, vaacaaq\}$$

cuya consulta (aa) contiene a los siguientes pares.

$$\{\langle 1,2,3\rangle, \langle 2,2,3\rangle, \langle 2,5,6\rangle, \langle 3,2,3\rangle, \langle 3,5,6\rangle\}$$

Si expandimos a derecha (aa) tenemos que calcular primero las siguientes consultas (aac)=$\{\langle 1,2,4\rangle, \langle 3,2,4\rangle\}$, (aab) =$\{\langle 2,2,4\rangle, \langle 2,5,7\rangle\}$ y $(aaq) = \emptyset$, calculando así la expansión $(aa)\cdot = \{\langle 1,2,4\rangle, \langle 2,2,4\rangle, \langle 2,5,7\rangle, \langle 3,2,4\rangle\}$. A partir de ahí podemos calcular las siguientes consultas: $(aa\rangle = \{\langle 1,2,3\rangle, \langle 2,2,3\rangle, \langle 2,5,6\rangle, \langle 3,2,3\rangle\}$ y $(aa] = \emptyset$. Expandiendo ahora a izquierda tenemos $\cdot(aa) = \{\langle 1,1,3\rangle, \langle 3,1,3\rangle\}$ junto con las consultas $\langle aa) = \{\langle 1,2,3\rangle, \langle 3,2,3\rangle\}$ y $[aa) = \{\langle 2,2,3\rangle, \langle 2,5,6\rangle, \langle 3,5,6\rangle\}$. Podemos ver ahora las consultas $[aa] = \emptyset$ y $\langle aa\rangle = \{\langle 1,2,3\rangle, \langle 3,2,3\rangle\}$. Aquí podemos ver que secuencia aa puede expandirse más a ambos lados.

Para introducir a una propiedad, calcularemos rápidamente $[vaac]$ y $[aab]$ en la figuras 3.2.1 y 3.2.2 respectivamente.

Las repeticiones $vaac$ y aab cuyas consultas son completamente no extensibles son las únicas para éste conjunto S. Lo importante es ver que éstas son repeticiones supermaximales del conjunto S. Más adelante demostraremos que la consulta de todas las secuencias completamente no extensibles da las tuplas correspondiente a las repeticiones supermaximales de un mismo conjunto de secuencias.

$$\begin{aligned}
(aab) &= \{\langle 2,2,4\rangle, \langle 2,5,7\rangle\} \\
(aab)\cdot &= \emptyset \\
\cdot(aab) &= \emptyset \\
(aab\rangle &= \emptyset \\
(aab] &= \{\langle 2,2,4\rangle, \langle 2,5,7\rangle\} \\
\langle aab) &= \emptyset \\
[aab) &= \{\langle 2,2,4\rangle, \langle 2,5,7\rangle\} \\
\langle aab\rangle &= \emptyset \\
[aab] &= \{\langle 2,2,4\rangle, \langle 2,5,7\rangle\}
\end{aligned}$$

FIGURA 3.2.2. Cálculo de la repetición supermaximal aab para el conjunto $S = \{vaac, waabaab, vaacaaq\}$.

A continuación presentamos unas propiedades de estos conjuntos.

COROLARIO 3.2.9. *La intersección entre una consulta extensible a izquierda (derecha) y una consulta no extensible hacia la izquierda (derecha) es vacía.*

$$\begin{aligned}
\langle\alpha) \cap [\alpha) &= \emptyset \\
(\alpha\rangle \cap (\alpha] &= \emptyset
\end{aligned}$$

DEMOSTRACIÓN. Como una tupla no puede pertenecer a una consulta extensible y a una consulta no extensible a la vez, la intersección es vacía. □

COROLARIO 3.2.10. *La extensión a derecha (izquierda) de una consulta no es extensible a derecha (izquierda) es vacía.*

$$\begin{aligned}
(\alpha]\cdot &= \emptyset \\
\cdot[\alpha) &= \emptyset
\end{aligned}$$

DEMOSTRACIÓN. La consulta no extensible a derecha (izquierda) por definición es el conjunto de los elementos que no pertenecen a la extensión a derecha (izquierda) de (α), entonces la consulta es vacía. □

COROLARIO 3.2.11. *La extensión de una consulta extensible a izquierda equivale a la extensión a izquierda de la consulta.*

$$\begin{aligned} (\alpha\rangle \cdot &= (\alpha) \cdot \\ \cdot \langle\alpha) &= \cdot (\alpha) \end{aligned}$$

DEMOSTRACIÓN. Como la consulta extendida modifica aquellas tuplas que pueden ser extendidas, la consulta extendida equivale a la consulta extensible extendida. □

La próxima consulta es de gran importancia ya que nos define lo que es un conjunto de secuencias supermaximales.

COROLARIO 3.2.12. *La consulta que contenga a todas las tuplas que correspondan a secuencias supermaximales para un conjunto S es igual a $[/\cdot *]$.*

DEMOSTRACIÓN. Tomemos un par maximal $\langle i, i', \pi, \pi', |\alpha|\rangle \in R(S)$ que corresponda a una repetición supermaximal $\alpha \in R'(S)$. En primera instancia α pertenece a la expresión regular $/\cdot */$. Por lo tanto las tuplas $\langle i, \pi, \pi + |\alpha|\rangle$ y $\langle i', \pi', \pi' + |\alpha|\rangle$ pertenecen a la consulta $(/\cdot */)$. Ahora por ser α una repetición supermaximal no se le puede colocar un símbolo a izquierda o a derecha sin que α deje de ser repetición, por lo tanto las tuplas deben pertenecerá también a $[/\cdot */]$. Hasta aquí demostramos que todos pares maximales pueden escribirse en tuplas en consultas y que estas pertenecen a la consulta no expansible $[/\cdot */]$.

Ahora tememos una tupla $\langle i, \pi, \pi + |\alpha|\rangle$ cualquiera de la consulta $[/\cdot */]$ correspondiente a la subsecuencia α. Por definición de una consulta por expresión regular sabemos que $[\alpha] \subseteq [/\cdot */]$, con lo que sin perder generalidad, podemos trabajamos con $[\alpha]$ en vez de $[/\cdot */]$. Justamente $\langle i, \pi, \pi + |\alpha|\rangle \in [\alpha]$. Esto significa que la tupla se repite más de una vez en S, por lo que existe otra tupla distinta $\langle i', \pi', \pi' + |\alpha|\rangle \in [\alpha]$, sino $[\alpha]$ sería una consulta vacía. A partir de estas dos tuplas podemos construir el par maximal $\langle i, i', \pi, \pi', |\alpha|\rangle$. Podemos decir esto ya que si $[\alpha]$ no es vacía entonces α no puede expandirse ni a izquierda ni a derecha sin dejar de ser repetición, por lo que podemos decir que α es una secuencia maximal, por lo tanto $\langle i, i', \pi, \pi', |\alpha|\rangle$ es un par maximal. Ahora, si existiera una consulta $[\omega\alpha\delta]$, con $\omega\alpha\delta \neq \alpha$ y no vacía, entonces la secuencia α no sería supermaximal. Pero como α no es extensible $\omega\alpha\delta$ no es una repetición. Gracias a que generalizamos sobre toda supersecuencia que contenga a α podemos concluir que no existe una repetición que la contenga, con lo que α es una repetición supermaximal. Podemos asegurar entonces que el par maximal $\langle i', \pi', \pi' + |\alpha|\rangle$ existe en $R(S)$. □

Esta demostración no solo demuestra que conseguir $[/\cdot*/]$ implica conseguir el conjunto de todas las copias de las repeticiones supermaximales, sino que también relaciona el álgebra de repeticiones con las definiciones de Gusfield. Para nuestro algoritmo necesitamos también ver que el conjunto de sufijos de $[/\cdot*/]$ equivale al conjunto de sufijos $(/\cdot*/]$, ya que esta es la base de la estructura de base propuesta para resolver el problema: árbol de sufijos de $[/\cdot*/]$.

COROLARIO 3.2.13. *El conjunto de sufijos de las repeticiones supermaximales de un conjunto S, equivale al conjunto de sufijos de las secuencias representadas por las tuplas de $(/\cdot*/]$.*

DEMOSTRACIÓN. Si α es una repetición maximal entonces $(\alpha] \subseteq (/\cdot*/]$, y $(\alpha] \neq \emptyset$. Por lo tanto todo sufijo de α esta incluido en los sufijos de las secuencias representadas por las tuplas de $(/\cdot*/]$. Ahora, sea α un sufijo de una secuencia representada por las tuplas de $(/\cdot*/]$. Hay que demostrar que existe un β tal que $\beta\alpha$ es una repetición supermaximal. Lo que sí podemos asegurar es que existe un γ tal que $(\gamma\alpha] \subseteq (/\cdot*/]$, $(\gamma\alpha] \neq \emptyset$. Es más, existe un γ tal que $\cdot\,(\gamma\alpha] = \emptyset$, con lo que $[\gamma\alpha] \neq \emptyset$. En consecuencia $\gamma\alpha$ es un supermaximal, en tanto α es prefijo de un supermaximal. □

3.3. Árbol de Sufijos de Supermaximales: AS^2

Resumiendo, se espera obtener un soporte que pueda almacenar todas secuencias repetidas supermaximales de un conjuntos de secuencias con un grado de repetición dado.

Las siguientes son características del modelo que se deben aprovechar para optimizar la consulta sobre el soporte.

1. La obtención de las secuencias es constante, por lo que es necesario que la inserción de los fragmentos no provoque un cuello de botella en el proceso de secuenciamiento.
2. El largo de las secuencias no supera un largo l. Se espera no tener más de m secuencias al final del secuenciamiento shotgun. El tamaño del alfabeto es de n símbolos.
3. Las repeticiones supermaximales son sobre un grado de repetición de c, que es la cobertura para tener el genoma terminado dado por la fórmula de Landerman.
4. La optimización debe centrarse sobre la consulta de repeticiones supermaximales.

Basándose en estos cuatro hechos, se propone el uso de un árbol de sufijos de las repeticiones maximales de un conjunto S de secuencias con un grado repetición de c.

Por las características propias de un árbol de sufijos **[12]** es conveniente para realizar consultas sobre cadenas (cadenas y subcadenas exactas e inexactas) permitiendo cumplir con el requisito 4. Pero no es obvio que se cumplan con los requisitos 2 y 3. A continuación presentaremos un algoritmo de inserción que nos permitirá realizar la inserción con una complejidad lineal al largo de la secuencia insertada, que construirá el árbol de sufijos de los elementos repetidos supermaximales de grado de repetición c. El largo máximo de las secuencias y la cantidad de fragmentos leídos al final del secuenciamiento nos permitirá calcular limites sobre el tamaño de la estructura.

El árbol de sufijos supermaximales tienen el siguiente invariante.

3.3.1. Invariante.

1. Cada camino ω del árbol que comience en la raíz representa un sufijo de una repetición supermaximal. Llamaremos $\overline{\omega}$ a la secuencia que corresponde al camino ω. Por definición de árbol solo existe un único camino a cada nodo del árbol. Por ello $\overrightarrow{\eta}$ es el camino entre la raíz y el nodo η, y $\overline{\eta}$ es la secuencia que corresponde al camino $\overrightarrow{\eta}$.
2. Cada nodo η del árbol contiene la consulta $(\overline{\eta}) - (\overline{\eta}\rangle$.

3.3.2. Propiedades.

1. Las hojas del árbol esta conformadas por $(\overline{\eta}) - \emptyset = (\overline{\eta}]$, ya que si existiera un hijo del nodo significaría que existe un sufijo de un supermaximal que lo continua, entonces $(\overline{\eta}\rangle \neq \emptyset$.
2. Las hojas del árbol cuyas consultas no pueden expandirse a izquierda representan repeticiones supermaximales.
3. La cantidad de subsecuencias almacenadas será de $m - n$ donde m es la cantidad de símbolos en total de las secuencias almacenadas, y n es la cantidad de secuencias almacenadas.
4. Si l es el tamaño de la repetición supermaximal más larga, entonces el árbol tiene l niveles, y menos de $|\Delta|^(l+1)$ nodos.
5. Todos los nodos tienen como mucho $|\Delta|c$ subsecuencias.
6. La cantidad de nodos es mayor a $\frac{m-n}{|\Delta|c}$.

Por ejemplo, la figura 3.3.1 muestra el árbol de sufijos de supermaximales para el conjunto de secuencias {CTGAC GTAA, CCCTT GGTAA A, TCGTA GTG, CATGC TTGGA} con un grado de repetición superior a uno. Cada rama del árbol

FIGURA 3.3.1. Árbol de sufijos de supermaximales del conjunto { *CTGACGTAA*, *CTGACGTAA*, *TCGTAGTG*, *CATGCTTGGA* } con un grado de repetición de 2.

representa cada uno de los sufijos de las repeticiones supermaximales: CC, CTTGG, CGTA, GA, GTAA.

3.4. Operaciones.

Insertar: Inserta todos los sufijos supermaximales de una secuencia al árbol. Para ello necesita de las siguientes operaciones auxiliares.

- **Insertar sufijo:** Inserta en el árbol el sufijo supermaximales que es prefijo de la secuencia de entrada dado por la operación insertar. Para completarse necesita de la operación de *llenar* y *expandir*.
- **Llenar:** Sea α prefijo de la secuencia de entrada tal que exista en el árbol una hoja h que le corresponda. Inserta en h la subsecuencia correspondiente a α. En el caso que no exista una hoja, inserta la subsecuencia correspondiente a α en el nodo correspondiente a la secuencia α.
- **Expandir:** A partir de un nodo h si existen más de c subsecuencias que continúan con el mismo símbolo s, se mueven estas subsecuencias al nodo correspondiente a la rama s. Si este nuevo nodo tienen más de c subsecuencias que continúan con un mismo símbolo, se aplica la operación expandir nuevamente.

Consultar por secuencia: A partir de una secuencia, consultar cuantas veces se repite en el conjunto y que secuencias la contienen. También puede indicar si es: *supermaximal*, *prefijo de supermaximal* y *sufijo de supermaximal*.

- **Todos los supermaximales:** Devolver los supermaximales almacenados en la base de datos.

3.4.1. Demostración que las operaciones mantienen el invariante. Las operaciones que modifican realmente el árbol son *llenar* y *expandir*, ya que *insertar sufijo* e *insertar* las usan para ingresar las secuencias. Las operaciones de consultas solo recorren el árbol.

Llenar: La operación de llenar rompe el invariante del árbol ya que agrega subsecuencias a las hojas sin fijarse si rompen el invariante de mantener

menos de c subsecuencias que continúen con el mismo símbolo. La operación *expandir* es quien vuelve válida a la estructura.

Expandir: Esta función vuelve al árbol a cumplir con el invariante. Cuando *llenar* agrega una nueva subsecuencia α al nodo, puede ocurrir dos cosas:

- Siendo s la secuencia que continua a α, la cantidad de secuencias que continúan con el símbolo s no superan el umbral c con lo que el árbol cumple con el invariante. *Expandir* no modifica el árbol. A este caso lo llamaremos *caso base* de la expansión.
- Siendo s la secuencia que continua a α, la cantidad de secuencias que continúan con el símbolo s supera el umbral c con lo que el árbol no cumple con el invariante. En este caso *expandir* mueve todos las subsecuencias que corresponden a α y continúan con el símbolo s al nodo correspondiente a la rama s. Con lo que este nodo, correspondiente a la secuencia αc, contiene las subsecuencias del nodo α que se pueden expandir a αc. A partir de aquí puede ocurrir nuevamente estos dos casos en este nodo y hay que repetir nuevamente la operación *expandir* hasta que termine en el caso base. A este caso lo llamaremos *caso recursivo* de la expansión.

Así queda demostrado de que estas operaciones mantienen el árbol dentro del invariante.

3.4.2. Ordenes de las operaciones.

3.4.2.1. Orden temporal de la inserción. La complejidad temporal de dichas operaciones es cuadrático, relacionadas directamente con el tamaño de la secuencia supermaximal más larga y el largo de la secuencia a inserción, siendo la menor de ambas la que tiene más peso. Veamos los cálculos para cada operación por separado empezando por la expansión. La implementación en donde están calculados estos tiempos corresponde a nodos con n listas enlazadas de tuplas, correspondiente a cada uno de los n símbolos del alfabeto. Tomaremos a l el largo de la secuencia a ingresar y m el largo de la secuencia supermaximal más larga.

- La expansión debe verificar por cada carácter si es posible expandir con el mismo, para eso hace compara la cantidad de subsecuencias que pueden expandirse con el mismo. La comparación tiene orden constante si se mantiene en la lista un contador de cantidad de elementos que guarda. Por lo tanto se realiza n comparaciones. Si se expande, solo se expande por una rama ya que si se tiene en cuenta que se insertó una tupla que desestabilizó el

nodo - superó el umbral de repeticiones -. La cantidad de tuplas que se van a mover son siempre $c + 1$, por la misma razón por la que se bajó por el nodo. Como esta operación se puede realizar hasta que no halla más que expandir, osea se terminó la secuencia o no hay repeticiones que superen el umbral, la cantidad de veces que se repite es el largo de la secuencia, l, o el tamaño de la maximal que comience en la posición indicada por la tupla expandida, que su cota será el largo de la secuencia supermaximal más larga, m. Por lo tanto, si sumamos la cantidad de comparaciones y movimientos en cada expansión tenemos $n + c + 1 + min(l, m)$ operaciones. Como n, y c son constantes tenemos una complejidad temporal de $O(min(l, m))$.

- La inserción en cambio tiene una complejidad cuadrática, ya que por cada secuencia hay que repetir la expansión l veces. Por lo tanto la complejidad de insertar una secuencia de largo l es de $O(l.min(l, m))$. Refinando, la secuencia en cada expansión es un carácter más corta, con lo que la complejidad se convierte en $O(\sum_{i=1}^{l} min(i, m))$. En el peor de los casos m va a superior a l con lo que la complejidad queda acotada a $\frac{l(l+1)}{2}$ por lo que el orden es cuadrático. En caso contrario el orden queda acotado por $\frac{m(m+1)}{2} + (l - m)m$.

Esta complejidad esta calculada para cualquier secuencia. ¿Pero cómo debería actuar este algoritmo en un modelo de secuenciamiento aleatorio? En principio todas las secuencias tienen en promedio un mismo largo (700pb), con lo que l se transforma en una constante. Por lo tanto esta acotado también m por no poder superar el tamaño de la secuencia más larga del conjunto. Entonces, aunque el algoritmo de inserción tenga un orden cuadrático, tiene que actuar en una complejidad temporal constante.

3.4.2.2. Ordenes temporales de las consultas. Con respecto a la identificación de secuencias supermaximales, se debe recorrer, en primera instancia todos los nodos del árbol hasta encontrar una hoja. Reconocer si la hoja es supermaximal es de orden lineal ya que la cantidad de elementos de un nodo no pude superar $n.c + s$, donde s es la cantidad de sufijos de las secuencias que terminan en aquel nodo. Si usamos este método para calcular el orden llegaremos a que es cuadrático, pero si tenemos en cuenta que la cantidad de tuplas que están almacenadas en todo el árbol es de $l. |S|$, donde $|S|$ es la cantidad de secuencias almacenadas y l es el largo promedio de todas ellas, y que recorrer todo el árbol es comparar todas las tuplas llegamos a la conclusión de que tiene un orden lineal: $O(l. |S|)$.

En el caso de que queramos verificar si una secuencia α es una repetición, el orden es lineal ya que solo tenemos que bajar por la rama que nos lleve al nodo n correspondiente a α, si existe. En caso contrario α no es repetición. $T(\alpha\ es\ repeticion) = |\alpha|$.

Además, si queremos obtener todas las secuencias que contienen a α debemos recorrer el subárbol donde el nodo n es su raíz. El tiempo de esta operación esta acotado a $T(\alpha\ es\ repeticion) + |\Delta|^{l-|\alpha|}$, ya que hay que recorrer como mucho esa cantidad de nodos para obtener las consultas de las secuencias que la contienen.

Decidir si una subsecuencia que esta una consulta es supermaximal tiene orden constante, ya que esta acotado como mucho por la cantidad de subsecuencias que puede tener una consulta (nodo) que por propiedad vimos que es de $|\Delta|c$. Primero se verifica que el nodo sea una hoja, sino es expansible a derecha, por lo que no es supermaximal. Luego se cuenta la cantidad de nodos continúan con cada símbolo para la izquierda. Si existe más de c subsecuencias para un mismo símbolo significa que es expansible a izquierda, por lo tanto no es supermaximal. En caso contrario se puede afirmar que es supermaximal para una cobertura de c.

Algorithm 1 Insertar: Inserta los sufijos supermaximales de S al árbol.

for all Sufijos ϵ de S **do**
  Llegar al nodo N correspondiente a un prefijo π de ϵ, tal que π es ϵ o N es una hoja del árbol. (*Llenar*)
  Agregar π a N.
  if π no es ϵ **then**
    Expandir el nodo N según el símbolo s, donde πs es prefijo de ϵ.
  end if
end for

Algorithm 2 Expandir: Expande un nodo N según el símbolo s.

if Cantidad de subsecuencias del nodo N que continúan con el símbolo s es mayor o igual a c **then**
  Mover todas las subsecuencias del nodo N que se continúan con el símbolo s al nodo correspondiente al eje s.
  for all Rama i del nodo M de la rama s **do**
    Expandir el nodo M según el símbolo i.
  end for
end if

CAPíTULO 4

Resultados y Conclusiones

4.1. Implementación

La implementación del AS^2 tiene como base un árbol n-ário, donde n es el número de símbolos del alfabeto. En nuestro caso son seis símbolos: ".ATCGN". El punto indica el inicio y el fin de la cadena. La "N" representa una incógnita en el nucleótido que correspondería a un lugar dado. Cada nodo tiene n listas enlazadas de subsecuencias. Las listas agrupan a las subsecuencias que tienen el símbolo próximo en común. Esto permite calcular la cantidad de elementos que tienen un mismo elemento próximo en orden constante, al igual que mover dichos elementos en la operación de *expandir*. Para reducir el espacio que ocupó el árbol se reduce todo nodo intermedio expandido en todas sus ramas. Cualquier subsecuencia que no pueda ser expandida a pesar que un nodo tenga todas sus ramas extendidas, significa que esa subsecuencia es una repetición maximal, pero no supermaximal. La estructura se almacena en memoria. Al final el proceso se pierde. Se sabe que puede almacenarse en disco y reusarse, pero se decidió dejar esta parte de la implementación en trabajos futuros. El lenguaje usado fue C++, y todos los procesos fueron ejecutados bajo un procesador Athlon de 1Ghz, con 256 Mb de memoria RAM.

4.2. Resultados

El objetivo de los experimento es confirmar el orden de temporal y espacial de los algoritmos. Luego observar el comportamiento del algoritmo en casos reales. Para ello fueron seleccionadas cuatro muestras. Las primeras dos muestras corresponden al genoma del *T. cruzi*, las otras dos al genoma del *Plasmodium falciparum*. El genoma del *T. cruzi* no se ha terminado de secuenciar por la gran cantidad de repeticiones que contiene, ver 1. Se ha seleccionado esta muestra no solo por que esta tesis intenta ser una punto de apoyo para solucionar los problemas de este parásito, sino también porque contiene dos elementos repetitivos complejos bien conocidos como son SIRE y VIPER [**19**]. Con la muestra VIPER/SIRE se intenta observar el comportamiento

de las repeticiones supermaximales con cobertura c con estructuras complejas como son estos dos elementos. Estos elementos fueron clasificados como *retrotransposones*. Además, el *T. cruzi* se esta secuenciando con una técnica de secuenciamiento aleatoria jerárquica. Esta técnica simplifica el ordenamiento fragmentando el genoma en n secciones del 100kb cada una, conocidas como BACs. Luego se secuencian cada BAC con la técnica shotgun. Luego se ensamblan los BACs por separados. Una vez resulto el ensamblado se termina de ordenar los BACs para conseguir un consenso que será el genoma del *T. cruzi*. Ordenar estos BACs se basa en organizar los *BACs ends*, extremos de 500pb de cada BAC. La segunda muestra se compone de las secuencias de estos extremos. Se quiere ver como reacciona el algoritmo a elementos superpuestos, pero que no representan la totalidad del genoma. Por último, a partir del genoma del *Plasmodium falciparum*, completado a través de la técnica WCS, se tomaron las siguientes muestras: lecturas de primera pasada del secuenciado aleatorio del cromosoma 2 y consenso del cromosoma 2 generado por el centro de investigación TIGR. Estas dos muestras nos permitirá ver el comportamiento del AS^2 en un caso real. Primero simulando la obtención de secuencias supermaximales durante el proceso de secuenciamiento, y segundo observando las secuencias supermaximales desde un genoma completo. Luego se podrá comparar ambos resultados para ver que tan buena es la predicción de repeticiones.

4.2.1. Confirmación de los ordenes. En todas las muestras se observaron que el tiempo y el espacio de la implementación confirman los órdenes del algoritmo como se ven en las figuras 4.2.1, 4.2.2, 4.2.3 y 4.2.4. Aún así, el orden temporal decae exponencialmente cuando el algoritmo usa memoria virtual. Esto nos deja suponer que el orden no empeorará si se usa el disco como soporte principal. Pero es un necesidad realizar un estudio minucioso en el algoritmo y estructura de datos que sera el soporte de la cache de la estructura, ya que el algoritmo usado por el sistema operativo para descargar una página de la memoria principal empeora el orden del algoritmo al punto de tener un comportamiento exponencial.

4.2.2. Observaciones extra. Más allá de los ordenes calculados, se ha observado un comportamiento lineal en el crecimiento de los nodos del árbol con respecto al crecimiento de los supermaximales, ver figuras 4.2.1, 4.2.2, 4.2.3 y 4.2.4. De la misma manera, y llamativamente, la cantidad de repeticiones supermaximales también tiene un comportamiento aproximadamente lineal con respecto a la cantidad de instancias ingresadas.

4.2.3. *Trypanosoma cruzi.*

4.2.3.1. VIPER y SIRE.

Origen de las secuencias: Entrez. Secuencias *T. cruzi* relacionadas con VIPER y SIRE. Búsqueda en Entrez:

```
("Trypanosoma cruzi"[Organism] AND
 (SIRE[All Fields] OR VIPER[All Fields]))
```

Se ha seleccionado dicho conjunto de secuencias para observar el comportamiento del modelo sobre un conjunto que contenga secuencias repetidas. Se quiere observar como una secuencia de la complejidad de VIPER se presenta bajo un modelo de repeticiones supermaximales exactas.

Cantidad de secuencias: 195.

Cobertura: 5,195. Exigimos dos coberturas, una muy restrictiva, como es 195 y otra menos restrictiva como es 5, para determinar que valor modela mejor las repeticiones.

Repeticiones: 12476 supermaximales con cobertura de 5x, 291 supermaximales con cobertura de 195x.

¿Como se observan a SIRE y VIPER?: En Entrez se puede encontrar a las SIRE y VIPER identificadas con los siguientes números gi2598052 y gi7248807 (VIP I) respectivamente. Se encontraron 168 supermaximales correspondientes a SIRE, y 1273 correspondientes a VIPER en los resultados con cobertura 5x. Las siguientes son las cinco repeticiones más significativas según el largo de las secuencias para SIRE[1]:

1. TTATTATTAT TATTATTATT ATTATTATTA TTATTATTAT TATTATTATT ATTATTATTA TTATTATTAT TATTATTATT ATTATTATT
   <gi2598052, 563, 653>, <gi2598052, 560, 650>, <gi2598052, 557, 647>, <gi2598052, 554, 644>, <gi2598052, 551, 641>, <gi2598052, 566, 656>,
2. CCTAACTTGT TTGGTTTCCA TAGATTATTT CAGGATCCGG CCAGCTGCCC
   <gi2598052, 51, 102>, <gi2598052, 489, 540>, <gi4323144, 368, 419>, <gi7248807, 3122, 3173>, <gi1679877, 2299, 2350>, <gi1911171, 761, 812>,
3. CCCCTTTTGA TTCCCACCAC GCGGCGGGGT CTTGTG
   <gi3366387, 142, 179>, <gi3366450, 367, 404>, <gi3392708, 134, 171>, <gi2598052, 308, 345>, <gi1673453, 489, 526>, <gi3258841, 368, 405>,
4. TTTTTATTTT GCCATCCCAC CCACCCCCT
   <gi3366285, 266, 296>, <gi1911171, 559, 589>, <gi2598052, 284, 314>, <gi4323144, 165, 195>, <gi624035, 991, 1021>, <gi7341240, 388, 418>,
5. TAGGAGCTTG TGAAAAGAAT GATCGCGG
   <gi1911171, 391, 420>, <gi2598052, 113, 142>, <gi3340683, 369, 398>, <gi1749835, 141, 170>, <gi3415244, 23, 52>, <gi8118103, 3587, 3616>, <gi1666218, 73, 102>, <gi4581861, 469, 498>,

[1]Cada secuencia se representa con la lista de subsecuencia, cada subsecuencia representada por tuplas de tres elementos, el primero es el nombre de la secuencia origen, el resto indican el inicio y fin de en la secuencia origen.

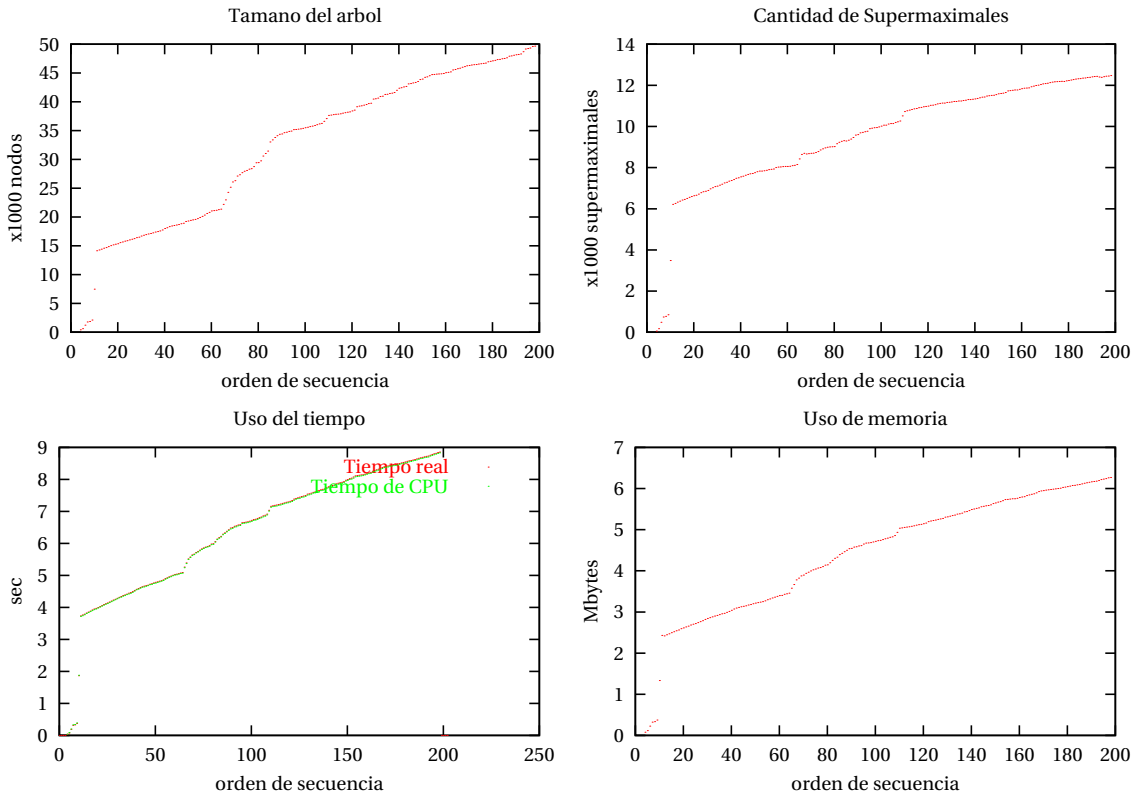

FIGURA 4.2.1. Resultados. Secuencias *Trypanosoma cruzi* con secuencias SIRE y VIPER. Cobertura 5x.

Con la cobertura de 195x se obtuvieron solamente 282 repeticiones pertenecientes a VIPER.Ninguna de SIRE.

4.2.3.2. BAC ends.

Origen de las secuencias: Entrez. Secuencias BAC ends. Las librerías BAC se componen de fragmentos de largos mayores a 100Kb. Se usan en secuenciamientos aleatorios jerárquicos. Es otra etapa del secuenciamiento donde es interesante observar la distribución de las repeticiones, ya que una gran cantidad de estas en esta etapa puede contraer grandes complicaciones en el ensamblado. En la actualidad este proceso se realiza con técnicas bioquímicas como *fingerprinting* para descartar los BACs que estén con estos elementos.

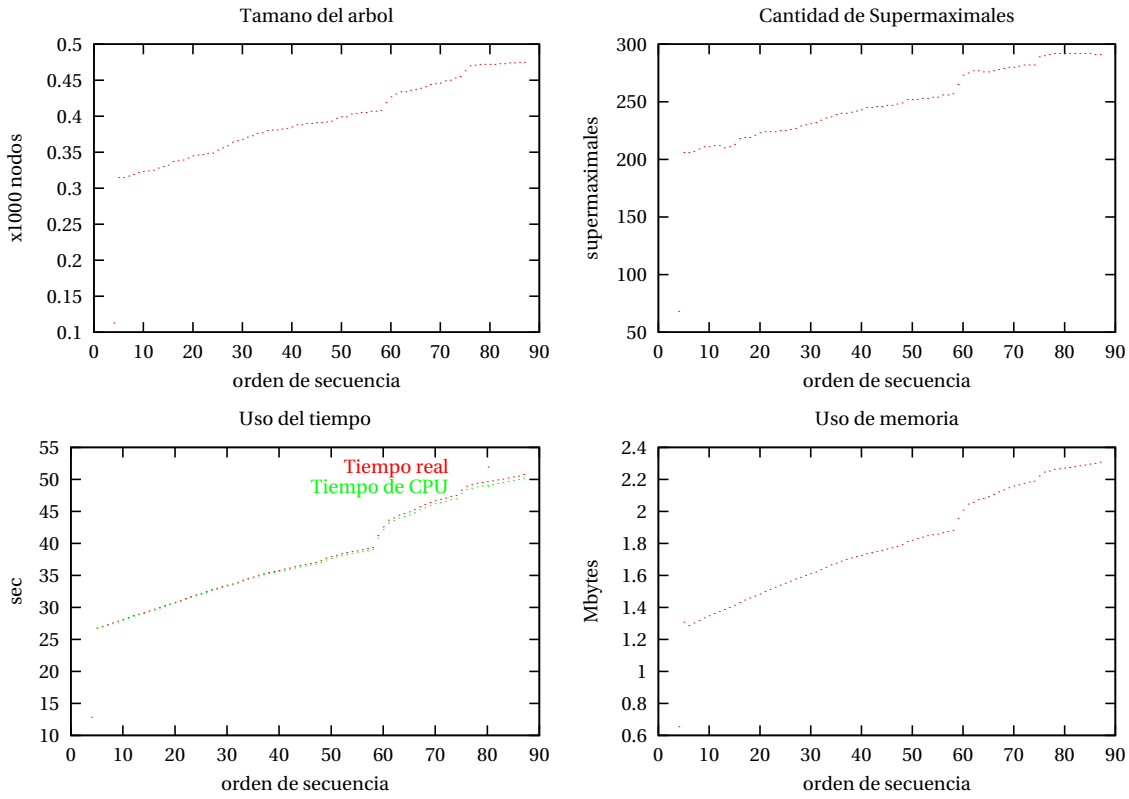

FIGURA 4.2.2. Resultados. Secuencias *Trypanosoma cruzi* con secuencias SIRE y VIPER. Cobertura 195x.

Ver "TIGR Sequencing Strategy for *T. cruzi*"en
http://www.tigr.org/tdb/mdb/tcdb/seq.shtml.
Consulta en Entrez:

```
("Trypanosoma cruzi"[Organism] AND BAC[All Fields])
```

Cantidad de secuencias: 14450.
Cobertura: 14. Este número es demostrativo.

4.2.4. *Plasmodium falciparum.*

4.2.4.1. Secuencias de primera lectura.

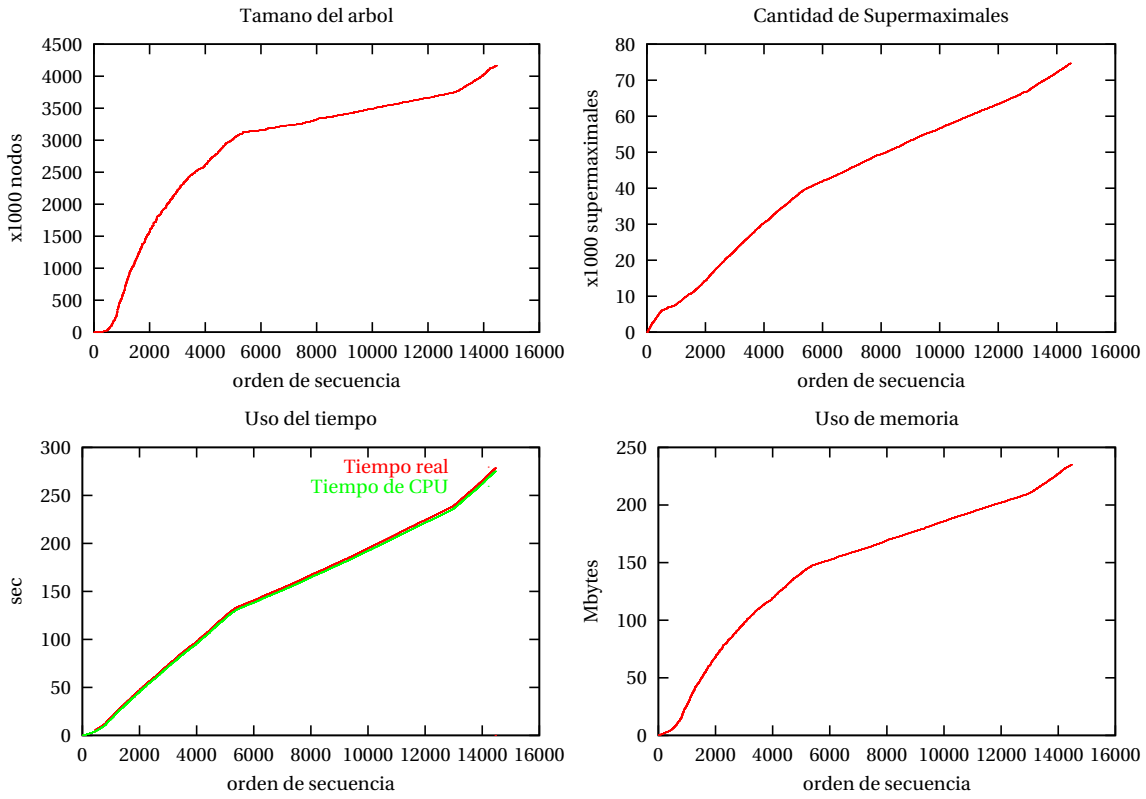

FIGURA 4.2.3. Resultados. Secuencias BAC ends del proyecto genoma del *Trypanosoma cruzi*, clon CL-Brener.

Origen de las secuencias: TIGR. Secuenciamiento WCS sobre el cromosoma 2 del *Plasmodium falciparum* con una cobertura de 10x.

Cantidad de secuencias: 21807

Cobertura: 20. Para encontrar únicamente repeticiones en un conjunto de secuencias cuya cobertura es 10x, se decidió acotar en 20 la cobertura del algoritmo.

4.2.4.2. Cromosoma 2 completo.

Origen de las secuencias: TIGR. Consenso del cromosoma 2 del *Plasmodium falciparum*.

Cantidad de secuencias: 1

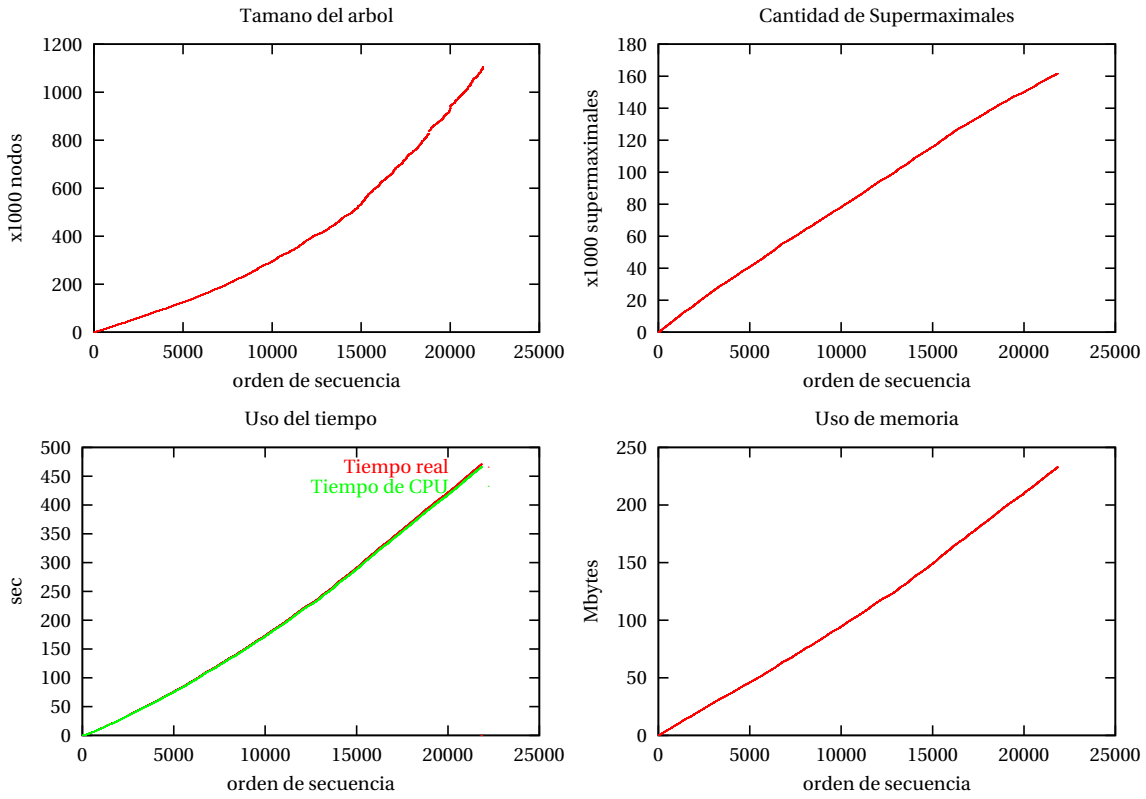

FIGURA 4.2.4. Resultados. *Plasmodium falciparum.*

Cobertura: 2. Como solo es una única secuencia, se tomo el valor 2 para encontrar todos las repeticiones exactas existentes en el cromosoma.

Luego de realizar una serie de experimentos con dichos algoritmos se ha observado la linealidad de los algoritmos. Se puede ver a continuación la lista de las pruebas con sus respectivos resultados. En el caso que la cantidad de nodos el árbol sea inferior a 100 podrá observarse también un gráfico del mismo. Se adjunta también comparaciones gráficas sobre el crecimiento de los nodos, cantidad de supermaximales, cantidad sufijos de supermaximales, memoria usada y tiempo de la inserción.

Muestras	tcruzi VIPER/SIRE	tcruzi BAC end	pfg chr2 WGS	pfg chr2 Completo
Secuencias	195	14450	21807	1
Cobertura	5	14	20	2
Tiempo (seg)	8,85	278,42	470,79	2819,31
Repeticiones	12476	74692	161619	32376
Nodos	49665	$4{,}17\text{x}10^9$	$1{,}10\text{x}10^9$	$99\text{x}10^6$
Memoria (Mb)	6,6	235,01	232,86	29,56

CUADRO 1. Resumen estadístico de las muestras seleccionadas.

4.3. Conclusiones

4.3.1. Árboles de sufijos de supermaximales. Mantener una estructura de un árbol de sufijos es muy difícil por el orden de almacenamiento que se necesita para una base de datos que se incremente diariamente con secuencias muy largas. Para almacenar el secuenciamiento de un genoma de s pb, con una cobertura de c, con la tecnología actual que nos permita obtener lecturas de promedio l, el árbol de sufijos necesita almacenar $c\frac{s}{l}$ fragmentos de un largo l. Con lo que nos queda que el árbol debe tener $c\frac{s}{l}\frac{l(l+1)}{2} = c\frac{s(l+1)}{2}$nodos. En un genoma como del *T. cruzi*, que tiene un tamaño aproximado de $87{,}10^6$pb, con una cobertura de 10, tenemos que necesitamos $28318, 5{,}10^7$. Si cada nodo se podría almacenar en un byte, necesitaríamos de $2{,}10^{11}$bytes, aproximadamente 18Gb.

4.3.2. Sobre el álgebra de repeticiones. En sí, el álgebra de repeticiones no da propiedades sobre las repeticiones. Solo fue definido para demostrar la validez del algoritmo. Aún así se han encontradas más propiedades que no son de utilidad para esta tesis, pero si que pueden ser de interés para futuros algoritmos sobre estos elementos.

4.3.3. Sobre el algoritmo y su implementación. La principal ventaja de este algoritmo no es la velocidad en que puede calcular todas las repeticiones maximales, sino la posibilidad de consultar dichas en cualquier momento del proceso de inserción, con lo que no es necesario tener terminado el proceso de inserción (secuenciamiento) para conocer todas las repeticiones. Así se puede encarar inteligentemente

el ensamblado de la secuencia principal, eliminando de antemano secuencias que sabemos que van provocar ambigüedades.

Es un detalle más que importante el uso de un árbol de sufijos como base de la estructura de datos. La cantidad de algoritmo de consultas de orden lineal que existen sobre dicha estructura da una potencia extra a la misma. Algunos de estos algoritmos son: alineamiento local, semi-global y global, búsqueda de palíndromos, etc.

Aún así, su uso esta limitado a la cantidad de memoria necesaria para procesar dicha información. Se espera poder implementar la estructura en disco para aumentar su capacidad, como también para reducir el efecto producido por el intercambio de páginas de memoria.

4.3.4. Sobre la cantidad de repeticiones supermaximales. La cantidad de repeticiones supermaximales exactas se hace muy complicado realizar un estudio serio sobre las repeticiones de un conjunto grande de secuencias. Por eso se recomienda el uso de visualización científica y data-mining que simplifiquen la tarea de obtener conclusiones sobre las secuencias.

Orden de importancia de secuencias. ¿Son importantes todas las secuencias repetidas supermaximales? La respuesta a esta pregunta es completamente subjetiva, por lo tanto dar una medida de importancia unánime es completamente irresponsable. Aún así se puede realizar un estudio estadístico sobre ellas viendo su complejidad, largo y cantidad de copias encontradas como primer punta del ovillo.

Visualización. La herramienta ideal para interpretar la gran cantidad de resultados de manera global es un gráfico. Existen varias formas de representar las secuencias repetidas y aquí trataremos de dar un par de ideas nuevas para su mejor interpretación.

- Grafos: Los grafos son una excelente manera de describir la composición de las repeticiones. Por ello, si es necesario estudiar la estructura de las secuencias, no hay mejor herramienta. La limitación mayor esta dada por la complejidad de los algoritmos sobre grafos que ayuden a realizar conclusiones sobre los mismos.
  - Secuencias relacionadas por supermaximales: Este grafo se basa en relacionar secuencias según los maximales que incluyen. Si visualizamos el grafo tratando de acercar a los nodos que más ejes tienen en común podremos visualizar nichos de secuencias donde las mismas están compuestas por una repetición/es más compleja/s. El defecto principal sobre

este grafo es la cantidad de nodos que usa dicho grafo, igual a la cantidad de secuencias de nuestro conjunto.

- Supermaximales relacionadas por secuencias: En vez de relacionar secuencias, ahora se relacionan supermaximales por secuencias. Esto solo reduce la cantidad de nodos, pero todavía sigue siendo difícil de manejar ya que la cantidad de ejes pasa a ser por lo menos la cantidad de secuencias de nuestro conjunto.
  - Orden de repeticiones: Una manera de ver la organización de las secuencias repetidas es ordenar las repeticiones según su ubicación dentro de las secuencias que las componen. Reduce la complejidad del grafo, aún así en conjuntos grandes se hace difícil de representar.
  - Orden de repeticiones sin relaciones transitivas: El grafo descripto anteriormente puede reducirse eliminándose los ejes transitivos. El defecto de este grafo es la complejidad del algoritmo para eliminar dichos ejes. Notar que no existen ciclos si tomamos como ciclo a aquellos caminos que repiten la misma copia de una repetición para una misa secuencia.

· Distribución de repeticiones sobre una secuencia: En vez de representar todas las secuencias en un único gráfico, puede visualizarse la distribución de las repeticiones por secuencia. La principal desventaja es la cantidad de secuencias que tiene el conjunto, por lo tanto se hace inútil obtener todas las secuencias representadas.

Navegación. En vez de tener un dibujo del resultado, se puede navegar permitiendo mostrar el resultado según la importancia que se quiera dar al problema. Nuevamente, hay que tener un punto de donde partir a buscar conclusiones, dar un orden de importancia a las secuencias se hace indispensable.

· Navegación basada en secuencias: Cambiar de secuencia gracias a la relación de secuencias por repeticiones comunes. El resultado es un grafo de secuencias relacionadas por repeticiones, pero filtrado por el usuario.

· Navegación basada en repeticiones: Según una repetición se puede relacionar con otra repetición según las secuencias que las contienen. El resultado es un grafo de repeticiones relacionadas por secuencias, pero filtrado por el usuario.

- A partir de un conjunto seleccionable de secuencias: En vez de obtener las repeticiones maximales a partir del conjunto completo, se puede elegir

un conjunto de secuencias y obtener cualquiera de los grafos descripto. Luego, eliminar o agregar secuencias hasta obtener un grafo deseado.

Reformular el concepto de secuencias supermaximales. El concepto propuesto de repetición no es suficientemente para un conjunto grande de secuencia. Además tiene un gran defecto, enmascara repeticiones que pueden tener un potencial muy grande para su estudio. Tomemos el caso de las islas CG; por la definición de repetición supermaximal, esta secuencia no es supermaximal si existe una secuencia maximal si la contiene, por lo tanto cualquier secuencia maximal que incluya la cadena CG hace que la repetición no sobresalga a la luz. Este problema puede resolverse con repeticiones inexactas o ampliando el concepto de secuencias supermaximales donde la cota de cantidad de copias no este dado por un número fijo, sino por un valor relativo. Por ejemplo, es maximal si se reduce por lo menos $\frac{1}{3}$ la cantidad de copias si se le agrega un nuevo carácter.

4.3.5. Resumen de las contribuciones.

Álgebra de subsecuencias: Existen propiedades del álgebra propuesta que no se ha desarrollado por salir del objetivo de la tesis, pero da un punto de vista interesante al tratamiento de elementos repetidos. En primer lugar puede ser la punta para desarrollar un lenguaje de consulta de este tipo de elementos, pero para ello primero se debe agregar más potencia de expresividad para incorporar, por ejemplo, repeticiones en tándem y repeticiones inexactas. También permitiría clasificar secuencias de forma unívoca. En si, es una herramienta nueva con mucho potencial en el área del estudio de repeticiones, como también en el desarrollo de algoritmos para su detección.

Nuevo modelo de repeticiones supermaximales: El modelo de repeticiones supermaximales en un conjunto de secuencias dependientes no fue planteado previamente. Creada en principio para resolver problemas relacionados con la genómica, al ser planteado para cualquier lenguaje, puede ser aplicada para la búsqueda de patrones en cualquier modelo donde exista un conjunto de muestras superpuestas hasta cierto grado de cobertura. Un ejemplo sería encontrar frases repetidas en un conjunto de trozos de hojas que provienen de copias de un mismo libro.

Árboles de sufijo de repeticiones supermaximales: La estructura AS^2 solo se especializa en supermaximales, con lo que no cae en la sobreinformación, y por basarse en una estructura muy estudiada como son los árboles de sufijo, se conoce muy bien sus propiedades.

Algoritmo para reconocer repeticiones: La aplicación que se le ha dado a la teoría desarrollada corresponde al secuenciamiento de genomas bajo una técnica de secuenciamiento aleatorio. El algoritmo desarrollado permite que durante todo este proceso, que dura años, pueda mantenerse una base de datos actualizada continuamente con la información de todas las secuencias repetidas del genoma según pase el tiempo. La consulta es inmediata y valida hasta ese momento, permitiendo seleccionar mejor las secuencias a ensamblar previniendo los problemas ya descriptos. También es una herramienta muy valiosa para quienes investigan estos elementos, ya que da una pauta de la cantidad y calidad de elementos repetidos existentes en el genoma.

4.3.6. Investigación futura.

Implementación de persistencia: La implementación usa la memoria principal para almacenar la estructura AS^2. Aunque se conoce que puede usarse el disco, aprovechando mejor la memoria principal y no llegar al problema de intercambio de páginas, no ha sido como objetivo de la tesis dicha implementación. También es de interés ver el comportamiento de la misma bajo un sistema concurrente, ya que en secuenciamientos a gran escala, donde más de cien máquinas aportan lecturas aleatorias durante todo el día, es necesario atender y organizar la mayor cantidad de información en el tiempo más corto posible.

Búsqueda inexacta de repeticiones: Como se ha descripto en la introducción, las repeticiones tienen regiones más y menos conservadas. El algoritmo propuesto solo busca regiones conservadas, sin términos medios, con lo que no se encuentras repeticiones en el concepto biológico. Es de interés agrandar el modelo a repeticiones inexactas para mejorar la calidad de la respuesta.

Nuevos modelos de repeticiones supermaximales: Como hemos visto, el concepto de maximal dependía de la inserción de un nuevo carácter. Aquí hemos agregado el concepto de cota en cantidad de copias. Sería de interés agregar un conceptos relativos, como por ejemplo el de porcentaje de copias con respecto a su largo.

Interpretación de los resultados: En esta tesis no se planteó observaciones sobre los elementos para poder caracterizarlos según su estructura. Es de interés poder clasificar de forma inteligente y unívoca estos elementos para su mejor estudio.

Visualización de los resultados: La cantidad de soluciones es extraordinaria. Un resultado interesante en el mar de supermaximales es casi imperceptible, incapaz de resaltar sobre los demás. Pero de la misma manera en que se puede observar grandes movimientos de aguas bajo la superficie del mar a grandes alturas, es necesario ver las diferentes formas en que se expresan estos elementos entre ellos. Por ello se propone realizar un intenso estudio sobre la visualización de los elementos para resaltar relaciones y propiedades que son imposibles de ver con solo conocer la secuencia.

Reensamblar subsecuencias de repeticiones: Existen técnicas para recuperar la historia de replicación de ADN [**9**]. A partir de este hecho podemos realizar dos preguntas que podrían resolver el problema de reensamblar subsecuencias de repeticiones: ¿Si se puede deducir dos historias parecidas de dos repeticiones, se puede deducir que pertenecen a la misma repetición? ¿La técnica propuesta por [**9**] es útil con fragmentos de repeticiones? También podemos preguntarnos que ocurre con la distribución y complejidad de estos elementos para saber si estos también son posible de relacionar. Hay un campo muy grande por donde investigar que puede ayudar a deducir repeticiones de gran tamaño, más grande de lo que soporta el secuenciado, y así resolver el problema de ambigüedad del ensamblado.

Bibliografía

[1] P. Agarwal and D. J. States. The repeat pattern toolkit (rpt): analyzing the structure and evolution of the c. elegans genome. *Proc Int Conf Intell Syst Mol Biol*, 2:1–9, 1994.

[2] V. N. Babenko, P. S. Kosarev, O. V. Vishnevsky, V. G. Levitsky, V. V. Basin, and A. S. Frolov. Investigating extended regulatory regions of genomic dna sequences. *Bioinformatics*, 15(7-8):644–53, Jul-Aug 1999.

[3] G. Benson. A space efficient algorithm for finding the best nonoverlapping alignment score. *Theoretical Computer Science*, 145(1&2):357–369, July 1995.

[4] G. Benson. Tandem repeats finder: a program to analyze dna sequences. *Nucleic Acids Res*, 27(2):573–80, Jan 15 1999.

[5] G. Benson and M. S. Waterman. A method for fast database search for all k-nucleotide repeats. *Nucleic Acids Res*, 22(22):4828–36, Nov 11 1994.

[6] compiled by Alan D. McNaught and Andrew Wilkinson. International union of pure and applied chemistry, compendium of chemical terminology: Recommendations. Blackwell Science, 1997. "Gold Book"6500+ definitions. http://www.chemsoc.org/chembytes/goldbook/.

[7] Wim M. Degrave. Report on the meeting of the trypanosoma cruzi genome initiative. Technical report, Trypanosoma cruzi genome initiative, April 1999.

[8] M. Ashburner et al. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(25):9, May 2000.

[9] W. M. Fitch, T. Smith, and J. L. Breslow. Detecting internally repeated sequences and inferring the history of duplication. *Methods Enzymol*, 128:773–88, 1986.

[10] E. Fredkin. Trie memory. *ACM*, 3(9):490–499, Sept 1960.

[11] Anthony J. F. Griffiths, William M. Gelbart, Jeffrey H. Miller, and Richard C. Lewontin. *Modern Genetic Analysis*. W. H. Freeman and Company, 1999.

[12] D. Gusfield. *Algorithms on strings, trees, and sequences.* Cambridge University Press, UK., 1997.

[13] H. V. Jagadish, Nick Koudas, and Divesh Srivastava. On effective multi-dimensional indexing for strings, 2000.

[14] E. W. Kannan, S. K.; Myers. An algorithm for locating nonoverlapping regions of maximum alignment score. *SIAM Journal on Computing*, 25(3):648–662, 1996.

[15] S. Kurtz, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. Giegerich. Computation and visualization of degenerate repeats in complete genomes, 2000.

[16] G. M. Landau, J. P. Schmidt, and D. Sokol. An algorithm for approximate tandem repeats. *J Comput Biol*, 8(1):1–18, 2001.

[17] E. S. Lander and M. S. Waterman. Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics*, 2(3):231–9, April 1988.

[18] M. Y. Leung, B. E. Blaisdell, C. Burge, and S. Karlin. An efficient algorithm for identifying matches with errors in multiple long molecular sequences. *J Mol Biol*, 221(4):1367–78, Oct 20 1991.

[19] M. Vazquez M., C. Ben-Dov, H. Lorenzi, T. Moore, A. Schijman, and M. Levin. The short interspersed repetitve element of trypanosoma cruzi, sire, is part of viper, an unusual retroelement related to long terminal repeat retrotransposons. *Proc. Natl. Acad. Sci. USA*, 97(5):2128–2133, 2000.

[20] U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. *SIAM J. Comput.*, 22:935–948, 1993.

[21] L. Marsan and M. Sagot. Algorithms for extracting structured motifs using a sux tree with an application to promoter and regulatory site consensus identication, 2000.

[22] E. McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23:262–272, 1976.

[23] D. R. Morrison. Patricia - practical algorithm to retrieve information coded in alphanumeric. *ACM*, 15(4):514–534, October 1968.

[24] NCBI. *DNA Sequences from Times Past in GenBank*, NCBI News, Spring 1999.

[25] NCBI. *The Human Genome Sequence: NCBI's First Annotated Edition*, NCBI News, Fall/Winter 2000.

[26] NCBI. *Contig Assembly and Annotation Process*, 2001.

[27] Leishmania Genome Network. Meeting report., 1999.

[28] U.S. Department of Energy. Human genome project information, oak ridge national laboratory. Part of the Primer on Molecular Genetics, June 1992. Genome Glossary.

[29] Richard Copley Peer Bork. Filling in the gaps. *Nature*, 409:218–820, Febrary 2001.

[30] E. Rivals, O. Delgrange, J. P. Delahaye, M. Dauchet, M. O. Delorme, A. Henaut, and E. Ollivier. Detection of significant patterns by compression algorithms: the case of approximate tandem repeats in dna sequences. *Comput Appl Biosci*, 13(2):131–6, April 1997.

[31] J. C. Roach. *Random Subcloning, Pairwise End Sequencing, and the Molecular Evolution of the Vertebrate Trypsinogens.* PhD thesis, University of Washington, 1998.

[32] Eugene Russo. Reading the human genome. *The Scientist*, 14(15):8, July 2000.

[33] J. P. Schmidt. All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *SIAM Journal on Computing*, 27(4):972–992, August 1998.

[34] David Parry-Smith Teresa Attwood. *Introduction to Bioinformatics.* AddisonWesley, 1999.

[35] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14:249–260, 1995.

[36] P. Vincens, L. Buffat, C. Andre, J. P. Chevrolat, J. F. Boisvieux, and S. Hazout2. A strategy for finding regions of similarity in complete genome sequences. *Bioinformatics*, 14(8):715–25, 1998.

[37] P. Weiner. Linear pattern matching algorithms. In *Proceedings of the IEEE 14th Annual Symposium on Switching and Automata Theory*, pages 1–11, 1973.

[38] Rocio C. Romero Zaliz. Reconocimiento y descripcion de objetos complejos en biologa molecular. Master's thesis, Universidad de Buenos Aires, 2001.