UNIVERSIDAD NACIONAL DE COLOMBIA

Aplicaciones de combinatoria sobre cadenas para analizar datos RNómicos y transcriptómicos de ARNts

Edgar Hernán Torres Baquero

Aplicaciones de combinatoria sobre cadenas para analizar datos RNómicos y transcriptómicos de ARNt

Edgar Hernán Torres Baquero

Tesis de grado presentada como requisito parcial para optar al título de:
Magíster en Ciencias Matemáticas Aplicadas

Director:
Clara Isabel Bermúdez Santana, Dr. Rer. Nat.
Co-director:
José Luis Ramírez Ramírez, Ph. D.

Línea de Investigación:
Combinatoria en palabras y RNómica
Grupo de Investigación:
RNómica Teórica y Computacional

Universidad Nacional de Colombia
Facultad de Ciencias, Departamento de Matemáticas
Bogotá D.C., Colombia
2017

Este trabajo se lo dedico a Isa, Miye, Pupe, Anita, Adry, Capullito.

Agradezco el apoyo y colaboración de Juan Gabriel Triana, Liliana Romero, Reinaldo Montañez, Clara Inés Sabogal, Martha Nelly Albornoz, Wilson López y Deicy Piedad Cañon. Y muy especialmente a mis directores de tesis Clara Isabel Bermúdez y José Luis Ramírez.

El tiempo se bifurca perfectamente hacia innumerables futuros

Jorge Luis Borges

Agradecimientos

Es difícil en un pocas líneas manifestar la inmensa gratitud que tengo, porque no solo me siento en deuda con personas, también con circunstancias, iniciativas, sentimientos, impulsos, ilusiones, miedos, todos ellos presentes a lo largo de este proceso y que me permitieron escribir párrafos de un trabajo representativo para mi historia de vida.

Quiero agradecer especialmente a la Universidad Nacional de Colombia, una versión a escala de nuestro querido país, donde tuve la oportunidad de conocer personas de diversas regiones, habilidades y culturas. Lugar que contrariamente a las creencias falsas, es un verdadero ejemplo de convivencia pacífica, tolerancia, trabajo duro y dedicación, donde se construye conocimiento y se trabaja a diario por demostrar el potencial que tiene la educación. Al servicio de intercambio Alemán DAAD y a la Facultad de Ciencias de la Universidad Nacional de Colombia Sede Bogotá por el equipamiento del laboratorio de Biologia Computacional de la Facultad de Ciencias, lugar en donde pude realizar la mayor cantidad de computo asociado.

Resumen

Basados en las investigaciones realizadas por el grupo de Transcriptómica de la Universidad de Leipzig y el Grupo de RNómica Teórica de la Universidad Nacional de Colombia, se ha demostrado la importancia de realizar análisis innovadores sobre bloques de alineamientos múltiples de ARNts. Estos bloques son construidos utilizando el programa *Blockbuster* y con su uso se ha logrado plantear un modo de caracterizar, preliminarmente las formas que toman los alineamientos sobre los ARNts.

Dada la creciente necesidad de desarrollar herramientas de cómputo para el estudio de bloques de alineamientos de lecturas sobre los ARNts, en esta tesis proponemos una metodología complementaria a la usada en *Blockbuster*. Esta nueva metodología esta basada en principios de combinatoria de palabras y busca, entre otras cosas, que pueda ser aplicada en diferentes tipos de moléculas biológicas, que desde la perspectiva de la bioinformática, describen particularidades de intrincados procesos celulares y tisulares sobre la morfología de los bloques de cadenas alineadas.

Con el objetivo principal de estudiar diferencias morfológicas entre estos alineamientos, para volúmenes significativos de secuencias alineadas, se utilizan algunos resultados de la combinatoria sobre cadenas de caracteres, más conocida como combinatoria en palabras, para el análisis de estos bloques de información.

En virtud de ello, esta tesis reúne una serie de conceptos orientados esencialmente al estudio de alineamientos.Para lo cual se definen relaciones entre cadenas de caracteres, caracterizando los bloques de alineamientos, que denominamos *cluster* y estudiando la complejidad de subcadenas y la valencia de cadenas, con el fin de presentar una implementación computacional, para ser aplicado sobre librerías de ARNts extraídas de tejido cerebral humano.

Palabras clave: transcriptómica, combinatoria, cadenas, clusters, alineamientos.

Abstract

Based on the investigations carried out by the Leipzig University's Transcriptomics group and the Universidad Nacional de Colombia's Theoretic RNómica Ggroup, the importance of performing innovative tests on multiple alignment blocks of tRNAs has been demonstrated. These blocks are built using the software Blockbuster and, with its use, it has been previously possible to propose a way of preliminary characterizing the forms alignments take on the tRNAs.

Using this thesis we are looking to propose a complementarty methodology to the one used in Blockbuster to characterize multiple alignment blocks of tRNAs given the increasing necessity in developing computational tools built on mathematical models based on combinatorial principles and the current boom in studies of alignment blocks of readings not only on tRNAs but on many other types of biological molecules that, from a bioinformatics' perspective, allow to identify peculiarities in intricate celluar and tissue processes influencing alignment blocks' morphology.

It is so, with the principal objective of studying morphologic diferences between these alignments, for significant volumes of aligned sequences, some results of combinatorial characters strings are used, better known as combinatorial on words to analyze these blocks of information.

Accordingly, this thesis brings together a number of concepts essentially oriented to the study of these alignments defining relations between characters strings, characterizing alignment blocks that we will call çluster.ªnd studying the total complexity and valence of chains, with the final purpose of proposing a computational implementation of the theoretical foundation particularly applied to libraries of tRNAs extracted from human brain tissue.

Keywords: transcriptomics, combinatorics, strings, words, clusters, aligns

Introducción

Desde el método inicial de secuenciamiento de cadenas de ADN o ARN propuesto por *Sanger* hasta los métodos actuales de secuenciamiento de *última generación*, encontramos que el trabajo computacional y el experimental estrechan cada vez más sus vínculos, no sólo con el objetivo de comunicar las disciplinas en los estudios genómicos sino también con el propósito de entender los intrincados procesos en los que están involucrados los ARNs. Es por esto que diversos grupos de investigación en bioinformática, orientan sus esfuerzos al estudio del transcriptoma, como se le denomina al conjunto de moléculas de ARN que cumplen funciones específicas y resultan de la transcripción de genes en las células.

Entre ellos encontramos el grupo de Transcriptómica de la Universidad de Leipzig y RNómica Teórica y Computacional de la Universidad Nacional, grupos que buscan desde las ciencias de la computación plantear alternativas analíticas en el estudio del transcriptoma. En este sentido, un resultado de gran impacto fue el diseño y desarrollo de las herramientas como *Segemehl* y *Blockbuster* usados para caracterizar bloques de alineamientos múltiples de secuencias de ARNs.

En esta tesis se desarrolla un procedimiento alternativo para el estudio de bloques de ARNs pequeños asociados con ARNts, mediante el uso de la combinatoria sobre cadenas y de las representaciones mediante cadenas de caracteres que se pueden hacer de sus contornos. En este texto, el lector encontrará inicialmente el contexto biológico, luego uno bioinformático en que se enmarcan los bloques de expresión de ARNs. Luego encontrará conceptos y resultados desde la combinatoria sobre cadenas de caracteres hasta su aplicación en la cuantificación y análisis sobre diversos tipos de ARNs pequeños alineados a ARNts de humano.

De acuerdo con lo anterior, esta tesis se reduce a la búsqueda de una metodología desde la combinatoria de palabras que puede servir como complemento a los resultados obtenidos desde la perspectiva estadística de *Blockbuster*.

Para comprobar algunos resultados de esta metodología se diseñaron algoritmos y se implementaron códigos en el lenguaje de programación *Python*, que luego fueron aplicados al análisis de una librería de ARNs, producto del secuenciamiento de ARNs derivados de células de tejido cerebral humano y que fueron mapeados al genoma humano y reclasificados en grupos de ARNts.

Los resultados de esta investigación, como el desarrollo de la metodología y aplicaciones computacionales, se llevaron a cabo en el laboratorio de Biología Computacional de la Facultad de Ciencias de la Universidad Nacional de Colombia Sede Bogotá.

Índice general

Índice de figuras

Capítulo 1

Contexto biológico y bioinformático

1.1. Contexto biológico: secuenciamiento de ácidos nucleicos

En las décadas de los sesenta y setenta, los científicos se esforzaron por desarrollar métodos para secuenciar ácidos nucleicos como el ADN o el ARN (DNA o RNA, por sus siglas en inglés) y proteínas, basados en un método químico en particular y cuyos productos, son denominados lecturas. El término lectura será denominado en los capítulos posteriores como cadenas o subcadenas dependiendo del contexto. Estas lecturas organizadas y acopladas por medio del usos de métodos computacionales permiten recuperar la posición y el orden de millones de caracteres que representan las cadenas de ADN o ARNs que en el lenguaje de la bioinformática se conocen como un conjunto ordenado de caracteres que representan las unidades básicas o nucleótidos (y que son representados en las secuencias de ADN o ARN por medio de los símbolos A, G, C y T o U). En particular, el tamaño de una secuencia es expresado en términos de pares de bases o bp. Ha sido tan extenso el programa de secuenciamiento de cadenas biológicas en el mundo que no es extraño hoy en día hablar de secuencias de genes, de proteínas, de cromosomas y de genomas en ámbitos tan diversos como en los biológicos, los bioinformáticos y los matemáticos.

Sin embargo, el secuenciamiento de las cadenas biológicas tiene una historia compleja que va desde el proceso químico usado en los procesos técnicos, hasta el diseño de algoritmos de alta complejidad que permiten recuperar el orden correcto de sus componentes hasta el desarrollo de máquinas robustas que permiten incrementar

el rendimiento computacional.

Para sorpresa de muchos biólogos moleculares, en el año 2005 se rompe con un paradigma al incorporar nuevas químicas de secuenciamiento que desplazan a los métodos clásicos de Sanger [33, 34]. Estos nuevos métodos, conocidos como métodos de secuenciación de última generación o HST, por su sigla en inglés *high-throughput sequencing*, hicieron que la biología se beneficiara con las mejoras en la cantidad, tamaño y eficiencia del proceso de secuenciamiento y que, como resultado, han generado toda una empresa industrial para el análisis de genomas y transcriptomas completos y por ende a su vez diversas áreas de investigación y campos de aplicación se han beneficiado en su desarrollo en la última década. Estos campos no solo incluyen los biológicos sino también los bioinformáticos y los de la ciencia de la computación, entre otros.

Con los años, se ha impulsado la demanda del incremento en el rendimiento para producir mayor número de productos de secuenciación, lo que ha conllevado al desarrollo incluso de una nueva era llamada la de la genómica personalizada. Este tipo de secuenciamiento ha transformado tanto el campo de la ciencia genómica y el de la biología en general, como se cita en [35], que son muchos los retos que desde el punto de vista bioinformático, algorítmico y computacional se deben afrontar. Cientos de trabajos han sido publicados en estos nuevos campos, que van desde datos cuantitativos sobre funciones biológicas específicas, hasta la caracterización de genomas de organismos importantes que son potencial para el uso sostenible de los ecosistemas.

Con el objetivo de descifrar los genomas de organismos completos, se hizo necesario un dramático aumento en el rendimiento e incremento de los capilares usados en la secuenciación del ADN. Este requisito se cumplió a través del impacto de la electroforesis capilar automatizada. Una gran revolución en el nuevo método de secuenciación del 2005, conllevo a la publicación de la secuenciación aleatoria y ensamblaje de novo del genoma de la *Mycoplasma genitalium* [26], conllevando a la necesidad de desarrollar nuevas soluciones computacionales para responder al incremento másivo en productos de secuenciamiento. Por primera vez las lecturas ya no son de tamaño de 1000 bp convensionales, sino que se empezaron a secuenciar en subcadenas que producían lecturas de tamaño variable desde 20 bp hasta 250 bp.

El laboratorio *George Church* desarrolló el protocolo llamado *Polony Multiplex* en el 2005, el cual fue utilizado por primera vez en el resecuenciamiento de una cepa evolucionada de *Escherichia coli*, con precisión de menos de un error por cada millón de nucleótidos [36]. Algunos estudios biológicos utilizan otros sistemas masivos

de secuenciación en paralelo, como el *1G Solexa* [2] y por medio de la inmunoprecipitación de cromatina surgió la metodología de secuenciamieno (*ChIPSeq*), y se realizaron otros ensayos al asignar la interacción proteína-ADN a través de todos los genomas de mamíferos [18].

Actualmente, las secuencias del ADN cromosómico humano, que pueden contener entre 55×10^6 y 250×10^6 bp pueden obtenerse en semanas. Sin embargo, el poner en orden las lecturas de su secuenciamiento para poder recuperar el ensamblaje de la secuencia nucleótica completa de un ADN cromosómico, es una tarea enorme computacionalmente, ya que las lecturas superan los millardos (ésto ya que es necesario fragmentar al azar, en pedazos muy cortos, el ADN cromosómico) que por complejos algoritmos implementados para correr sobre máquinas de cómputo de alto rendimiento, son "sobrelapadas para lograr recuperar o ensamblar el orden inicial de la secuencia original.

Paralelo al secuenciamento del ADN a escala rápida, también surgió la secuenciación de los ARNs, técnica conocida como ARN-seq o RNA-seq por su nombre coloquial en inglés. Una herramienta revolucionaria, usada para descifrar la complejidad del funcionamiento de los genomas, es decir, ya no solo se pretende conocer cual es la secuencia del genoma de una especie, sino que además hoy en día, se pretende secuenciar todas las moléculas existentes de ARNs transcritas desde los genomas y que son directamente asociadas como productos de expresión de los genes. A este conjunto de moléculas tipo ARN se conoce como transcriptoma que es también conocido como el conjunto de todas las moléculas funcionales de ARNs que son producto de la transcripción en una célula o en una población de células [41]. Estos métodos combinan la tecnología *HTS* y la medición cuantitativa de las transcripciones y sus formas alternativas (isoformas), ver [41]. Entre otras cosas la técnica *ARN-seq* ha permitido:

- Cuantificar los transcriptomas de mamíferos [29].
- Caracterizar los transcriptomas de las células madre y reconstruir el perfil del transcriptoma de células modelo para estudiar el cáncer, como las células HeLa, S3 [6, 27].
- Tipificar la transcripción del genoma de la levadura [30].
- Desarrollar mapas de alta resolución integrados al epigenoma en Arabidopsis [24].
- Encontrar propiedades del transcriptoma mediante reordenamientos genómicos en una línea de células de mama cancerosas [43].

- Caracterizar el transcriptoma de muchos tejidos asociados con enfermedades humanas.
- Crear nuevas áreas dentro de la genética, entre las que se encuentra la identificación de promotores alternativos, utilizados para la producción de otros transcritos, ver [37].

Con sólo cambios menores, las técnicas se extendieron a laboratorios de todo el mundo, y sentaron las bases para la revolución en la secuenciación de las décadas de los ochenta y noventa, con el subsiguiente desarrollo, dentro de la bioinformática, de las ciencias ómicas las cuales dominan y cautivan investigadores de diversas áreas.

1.2. Caracterización de ARNs no codificantes utilizando secuenciamiento profundo de última generación

Los ARN no codificantes son moléculas de gran interés desde el punto de vista biológico debido a las múltiples funciones que realizan en la célula, tales como:

- Estructurales, como los ARN ribosomales que forman los ribosomas o complejos macromoleculares, donde se da la síntesis de proteínas.
- Catalíticas, como las funciones que son realizadas por enzimas de RNA o llamadas Ribozimas.
- De procesamiento o splicing, como los ARNs nucleares (o snRNA en inglés).
- Como adaptadores de aminoácidos los ARNs de transferencia ARNts o (tRNAs por su sigla en inglés).
- De regulación de la expresión de genes como los miARNs o mas globalmente conocidos como los miRNAs por su sigla en inglés.

Por ésto, la última década ha sido marcada por la necesidad de identificar nuevas moléculas que conforman el transcriptoma y de medir nuevas funciones para ellas, como el conocido nuevo modo de silenciamiento de genes mediados por ARNs llamado ARNi del inglés o (RNAi o RNA interferente), el descubrimiento de nuevos ARNs no codificantes como los *Vault* ARNs. Esta decada ha requerido del desarrollo de nuevos métodos computacionales igualmente al paralelo del desarrollo de los

nuevos métodos experimentales de secuenciamiento que han acompañado la prolifera "caza de nuevas moléculas.

Por ello, actualmente, se dispone de bibliotecas de ARNs pequeños, también producto de la transcripción y que han sido secuenciadas utilizando ARNseq y a su vez de herramientas de cómputo capaces de su análisis. Estas técnicas computacionales se conocen como mapeo y se basan en la búsqueda de máximas coincidencias entre cadenas comparables. Muchas de estas metodologías son parte de los métodos de trabajos pioneros que han podido acoplarlas a sitios no canónicos en los genomas y también han favorecido la búsqueda de posibles patrones medibles sobre los inmensos volúmenes que pueden ser secuenciados en un sólo experimento biológico.

En palabras de Schuster, hoy por hoy los científicos de la computación y los experimentalistas han respondido al desafío de trabajo interdisciplinario para crear nuevas y eficientes herramientas que permitan avanzar en el área [35], aunque aún resta mucho por desarrollar soluciones que permitan comparar patrones generados desde los volumenes de productos de secuenciamiento o perfiles de expresión, como se pretende abordar en esta tesis.

1.3. ARNts, patrones de expresión y procesamiento

Un ejemplo de la diversificación de funciones de ARNs no codificantes lo constituye el ARNt o comunmente llamado tRNA. En el 2010, gracias a los datos generados por ARNseq fue posible identificar que los clásicos adaptadores de aminoácidos en el proceso de traducción, son fuente generadora de otro tipo de moléculas llamados fragmentos derivados de ARNts [24].

La historia de su descubrimiento se relaciona inicialmente con los análisis de productos derivados de la biogénesis de ARNts y productos derivados del rompimiento o fragmentación de sus cadenas, como respuesta a cambios ambientales. Estos productos de fraccionamiento pueden ser derivados como respuesta a estrés celular, cambios en el desarrollo de un organismo, cambios en la estabilidad estructural de ARNts y otros procesos biológicos [23, 39, 38, 42, 15, 12, 17, 5, 9, 31]. Sin embargo, a inicios del 2010, basados en análisis total de transcriptoma, se identificaron nuevos ARNs pequeños derivados de ARNts [7, 22, 13, 11]. Secuenciamiento profundo de extractos celulares revelaron que los ARNs pequeños más abundantes son productos de procesamiento de los ARNts de lisina, valina, glutamina y arginina. Estos ARNs son especialmente procesados desde el extremo 5' de los ARNts por la

enzima *Dicer* (que fragmenta sobre el D-loop del ARNt), enzima muy importante para procesar ARNs de interferencia, produciendo ARNs pequeños de aproximadamente 19 nucleótidos de tamaño, ver [7]. Es importante mencionar que los ARNts se clasifican en 20 tipos según el aminoácido que transportan y que llamaremos como grupos codónicos o grupos codones, para conocer los diferentes grupos se puede consultar la Tabla A.

Basados en el análisis de perfiles de expresión global de ARNs pequeños de líneas celulares de cáncer de próstata humana, tres tipos de fragmentos han sido identificados: *tRF-5*, *tRF-3* y *tRF-1* [22]. Hallazgos recientes proponen que estos fragmentos pueden competir con procesos del silenciamiento mediado por ARNs en células de mamíferos [13].

Sin embargo, el análisis de datos de perfiles de expresión de los ARNts muestran variedad, la cual aún no ha sido caracterizada y aún se mantienen, preguntas abiertas al respecto, como las de determinar si existe alguna conservación en las secuencias de las cuales procedan los ARNs pequeños o si existe alguna correlación entre estas secuencias y las subregiones de los ARNts que puedan ser generadoras de ARNs pequeños [10].

1.4. Contexto bioinformático: mapeo de lecturas del secuenciamiento a los genomas de referencia

1.4.1. Mapeo de secuencias: *Segemehl*

Con el fin de poder realizar el mapeo de ARNs pequeños al genoma humano, en esta investigación utilizamos el software *Segemehl*, que tiene como propósito el mapeo de secuencias pequeñas a regiones genómicas de referencia. Este software permite identificar no solo errores de mapeo en el secuenciamiento, sino que también permite conocer borramientos e inserciones de caracteres que ocurren en el proceso técnico del secuenciamiento. Este método se fundamenta en algoritmos basados en árboles de sufijos mejorados, ver [14].

1.4.2. Búsqueda de bloques de expresión: *Blockbuster*

Como se puede observar, el estudio del transcriptoma requiere de complejos procesos de cómputo, con el fin de determinar si diversas funciones celulares están

determinadas por el comportamiento y propiedades de los alineamientos múltiples de cadenas cortas de ARN.

En razón a lo anterior, distintos grupos de investigación están desarrollando herramientas para la caracterización de los bloques o acumulaciones de cadenas alineadas a determinados subregiones de los ARNts. Entre estos desarrollos, se destaca `Blockbuster`, ver [20] y [4], con el cual se han logrado encontrar distintos patrones de expresión vistos como diversos bloques de acumulación de lecturas de ARN pequeños a los diversos grupos de ARNts.

En este sentido esta tesis se presenta como una propuesta alternativa para el estudio de contornos derivados de los bloques de cadenas cortas, obtenidas previamente por el uso de *Blockbuster* que es nuestro referente, y que nos permite mostrar en el capítulo 3, que la metodología que se plantea, mediante la combinatoria sobre cadenas, representa otra caracterización práctica y reproducible de los bloques.

A continuación se presentan las definiciones, respetando la notación literal de los autores y su propiedad intelectual en la idea de bloques o bloques de expresión.

Definición de Cluster Las lecturas mapeadas son ordenadas por posición genómica. Dadas dos lecturas estas son asignadas a un cluster siguiento un criterio de asignación de la lectura a un cluster si su distancia es menor o igual a 39 nucleótidos y siguiendo las siguientes definiciones:

Definición 1: Una cadena s es una lectura, es decir una secuencia que en la librería de secuencias comprimidas fue secuenciada. Toda s tiene una longitud l, $17 \leq l \leq 28$ donde $l = 17$ ($l = 28$) representa el mínimo (máximo) longitud de cadena. Es decir que toda s tiene un punto de inicio a y un punto final b, entonces s es representado por (a, b) y $l = b - a + 1$.

Definición 2: Dado un conjunto de cadenas ordenadas $C = \{s_n\}_n$, $(s_n = (a_n, b_n))$, se define la distancia δ_{n+1} entre dos cadenas consecutivas s_n, s_{n+1} como $\delta_{n+1} = a_{n+1} - b_n$.

Definición 3: Un *cluster* es un conjunto ordenado de cadenas $C = \{s_n\}_n$ (con al menos 10 elementos de s_n) tales que la distancia δ_{n+1} satisfaga que $\delta_{n+1} \leq 39$ para $s_n \in C$. Esta medida de 39, se obtiene añadiendo 11 (la diferencia máxima entre cadenas) y la longitud máxima de una cadena. Ver figura 1.4.2 para observar una abstracción del modelo

Blockbuster Una vez definidos los *loci* (o posiciónes de mapeo en el genoma de referencia) de ARNs, se enfrenta el problema de dividir lecturas consecutivamente en bloques para detectar patrones que son asociados a la expresión específica de la región del genoma que provienen. Hay que tener en cuenta que esta tarea es diferente a la de segmentación, por ejemplo, a los perfiles de teselados [16]. Ya no es posible *a priori* restringir a los bloques que no se superponen. Debido a la variabilidad y a las imprecisiones en la secuenciación, no siempre los arreglos de lecturas muestran fronteras exactas. Por tanto, *blockbuster* se presentan como una herramienta automática que reconoce los bloques de lecturas. En primera instancia, una lectura mapeada u con posiciones inicial y final, a_u y b_u, respectivamente, se sustituye por la densidad normal gaussiana ρ_u con media $\mu_u = (b_u + a_u)/2$ y varianza σ_u^2. Tenemos que $\sigma_u = s|(b_u - a_u)/2|$, donde s es un parámetro que se utiliza para ajustar la resolución. Para cada *locus*,(o posición única de mapeo) estas densidades gaussianas se agregan separadamente para las dos direcciones de las lecturas, obteniendo las curvas f^+ y f^- que evidencian suaves y pronunciados picos, centrados en los bloques de lecturas, con puntos medios casi idénticos que de forma abstracta modelan los bloques observados en la figura 1.4.2. Ahora bien, en este punto, usamos un procedimiento exhaustivo para extraer las lecturas que pertenecen al mismo bloque:

1. Determinar la ubicación $\hat{x}$ del pico más alto.
   Sea $B = \emptyset$ y $\delta = 0$.

2. Incluir en el bloque B todas las lecturas u tales que
   $\hat{x} \in [\mu_u - (\sigma_u + \delta), \mu_u + (\sigma_u + \delta)]$.
   Sea δ la desviación estándar de μ_u, $u \in B$ y se repite el paso hasta que no se incluyan lecturas adicionales u.

3. Calcular $f_B = \sum_{u \in B} \rho_u$, luego remueva las lecturas en B, y asignar $f \to f - f_B$.

Este procedimiento iterativo extrae bloques en su correspondiente orden de importancia como se observa en la figura 1.4.2. Dado que el área comprendida en la región debajo del pico es igual al número de lecturas del bloque, la altura representa la expresión total de un bloque.

Por lo tanto, es pertinente usar la altura del pico para definir una condición de parada de *Blockbuster*. Se usa $s = 0{,}5$, un valor necesario para que los bloques queden separados, con el fin de ser reconocidos como distintos. Es importante resaltar que

este procedimiento de detección de bloques puede ser realizado usando la aproximación mediante deconvolución Gaussiana, la cual es comúnmente utilizada, por ejemplo, en cromatografía [40] y muchas áreas de espectroscopia.

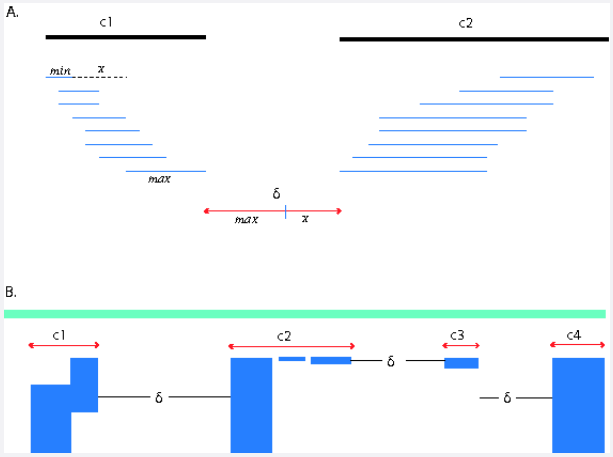


Figura 1.1: Los bloques de lecturas o de ARNs pequeños son modelados usando el método descrito en el cual cada lectura se sustituye por la densidad normal gaussiana. En la figura por ejemplo, un *cluster* llamado por simplicidad c1 es un conjunto ordenado de cadenas $C = \{s_n\}_n$ (con al menos 10 elementos de s_n) tales que la distancia δ_{n+1} satisfaga que $\delta_{n+1} \leq 39$ para $s_n \in C$. En rojo se representa el tamaño de cubrimiento del cluster sobre la subregion genómica (en color verde). Las cadenas o lecturas son representadas en A como lineas azules delgadas y en B como bloques construidos a partir del método de *Blockbuster*. Tomada de [4]

.

Capítulo 2

Cadenas y alineamiento de subcadenas

2.1. Cadenas

La definiciones y propiedades que se presentan a continuación son el fundamento matemático que soporta el modelo presentado en el Capitulo 3, y son derivadas de [25].

Sea Σ un alfabeto o conjunto finito de elementos llamados caracteres. Se define la operación concatenación como $a \cdot b = ab$ con $a, b \in \Sigma$ y se denotada por $\cdot$. Entonces, llamaremos cadena w a la concatenación de elementos de un alfabeto Σ. La longitud $|w|$ corresponderá al número de caracteres que componen la cadena w. Se resalta que en este documento solo trabajaremos con cadenas finitas, es decir aquellas que tienen una cantidad finita de caracteres concatenados.

Además, denotaremos como λ a la cadena vacía, y a Σ^* como el conjunto de todas las cadenas formadas por símbolos Σ incluyendo a λ y al conjunto $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. El conjunto Σ^n representa al conjunto de todas las cadenas de longitud n, con $n \in \mathbb{N}$. Además, resaltamos que $\Sigma^* = \bigcup_{i \in \mathbb{N}} \Sigma^i$. Note por ejemplo que la dupla $(\Sigma^*, \cdot)$ es un monoide porque es asociativo y λ corresponde al módulo para la operación concatenación.

Por otro lado, llamaremos factor de w a la cadena u, si existen cadenas x, y tales que $w = xuy$. Ahora bien, en el caso en el que $x = \lambda$, denominaremos al factor u como *prefijo*, y si $y = \lambda$ entonces llamaremos a u como *sufijo*. La expresión $w[0 \dots k-1]$

corresponderá al prefijo de longitud $k \in \mathbb{N}$ y $w[n-k \ldots n-1]$ al sufijo de longitud k.

Es importante resaltar que en el estudio de las cadenas de caracteres existen algunos órdenes que son utilizados de manera habitual y por lo tanto en este punto introducimos algunos de ellos:

- **Orden prefijo:** $u \preceq_p v$ si existe un $w \in W$ tal que $v = uw$.
- **Orden de subpalabra:** $u \preceq_i v$ si existe un $w, t \in W$ tal que $v = wut$.
- **Orden sufijo:** $u \preceq_p v$ si existe un $w \in W$ tal que $v = wu$.
- **Orden lexicográfico:** decimos que $u \preceq_l v$ si $u \preceq_p v$ o que existe una factorización $u = xau'$ y $v = xbv'$ con a, b caracteres tales que $a < b$ (o que están bajo el orden de los caracteres en el alfabeto).

Ejemplo: Para las cadenas u =AAGGC, v =AAGGCGAGAAGGCGAG y z =GGCGAG, tenemos que:

- **Orden prefijo:** $u \preceq_p v$, dado que $v = uw$, donde w =GAGAAGGCGAG.
- **Orden de subpalabra:** $z \preceq_r v$, dado que se puede expresar $v = tzr$ donde t =AA y r =AAGGCGAG.
- **Orden sufijo:** $z \preceq v$, dado que si s =AAGGCGAGAA, v se puede expresar como $v = sz$.
- **Orden lexicográfico**: $u \preceq_l v$ dado que $u \preceq_p v$. Ahora $v \preceq_l z$ dado que al factorizar $v = \lambda A u'$ siendo u' =AGGCGAGAAGGCGAG y $z = \lambda G z'$ con z' =GCGAG, $A < G$. □

Estos ordenes nos serán de gran utilidad en el capítulo 3, dado que requerimos establecer varias relaciones de orden para hacer el procesamiento de los bloques de alineamientos de secuencias de ARNs, al igual que para ordenar las subcadenas o factores contenidas en una cadena de caracteres.

2.2. Árbol de palabras clave

El problema del mapeo de cadenas a una cadena de referencia depende principalmente de la búsqueda de patrones o subcadenas de una cadena determinada. Estos patrones en nuestra aproximación, son considerados como factores, y los llamaremos de esta manera, con el fin de poder relacionarlos con el proceso de alineamiento

y mapeo de cadenas a cadenas de referencia.

Inicialmente, hablaremos del proceso de emparejamiento de múltiples patrones, ver [19]. Sea w una cadena de longitud n, y sean $p_1, p_2, \ldots, p_k$ patrones. Ahora podemos expresar a w como $w = w[0]w[1]\ldots w[n-1]$. El problema principal en el emparejamiento múltiple de patrones se resume en a encontrar las posiciones de inicio de los patrones en la cadena w.

Para ello, Alfred Aho y Margaret Corasick propusieron en el año 1975, ver [1], la definición del *árbol de palabras clave*, que formalmente corresponde a un árbol enraizado que satisface las siguientes condiciones. Asumimos por simplicidad que ningún prefijo en el conjunto es un prefijo de otro patrón:

- Cada arista del árbol es etiquetada con una letra del alfabeto.
- Cualquier pareja de aristas que salga de un mismo vértice tiene etiquetas distintas.
- Todo patrón p_i, $(1 \leq i \leq k)$, de un conjunto de patrones es deletreado en algún camino del *árbol de palabra clave* desde la raíz a una hoja.

Este árbol puede ser usado con el fin de encontrar la posición de inicio de los patrones $p_1, p_2, \ldots, p_k$ en una cadena.

Ejemplo: En la Figura 2.2 se muestra el árbol de palabras clave de la cadena *GATGAC*.

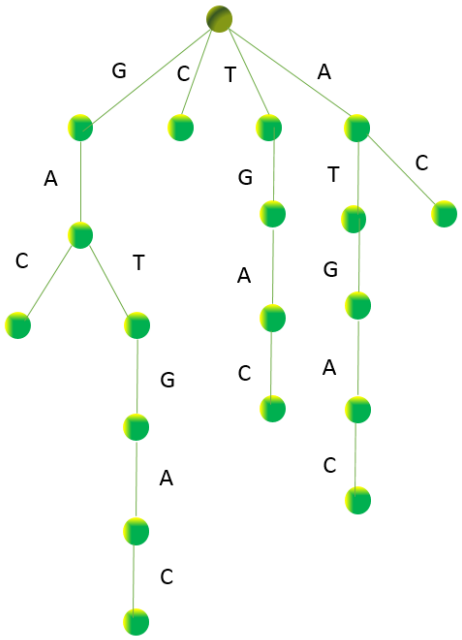


Figura 2.1: Note que cada arista del árbol ha sido etiquetada con una letra del alfabeto y desde cualquier vértice interno del árbol las aristas generadas están etiquetadas de los letras diferentes. Por ejemplo el patrón GAC se deletrea sobre el camino construido desde el nodo raíz de la primera hoja del árbol. La hoja corresponde a la primera desde la izquierda a la derecha del árbol

2.3. Árbol de sufijos

Un *árbol de sufijos* de una cadena $w = w[0]w[1]\dots w[n-1]$ es enraizado y está etiquetado con n hojas (numeradas de 0 a $n-1$) si satisface las siguientes propiedades, ver[19]:

- Cada arista es etiquetada con una subcadena de w.
- Cada vértice externo (exceptuando posiblemente la raíz) tiene al menos dos hijos.
- Cualquier pareja de aristas que salga de un mismo vértice tiene etiquetas que comienzan con un carácter distinto.

- Todo sufijo de w es deletreado en un camino de la raíz a las hojas.

Ejemplo: En la Figura 2.3 se ilustra el árbol de sufijos de la cadena $w = GATGAC$.

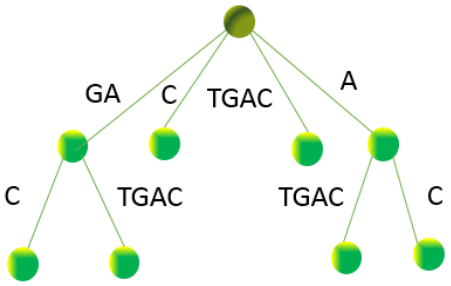


Figura 2.2: Note que las aristas son etiquetadas con subcadenas de w, y de los vértices internos se derivan al menos dos hijos y las subcadenas que etiquetan las aristas comienzan con caracteres distintos

2.4. Complejidad de cadenas

En esta sección estudiaremos una noción de complejidad de cadenas de caracteres, la cual está basada del artículo [8] en un contexto aplicado directamente sobre análisis de cadenas de ADN. Este contexto se incluye con el propósito de poder analizar las cadenas que describen los contornos de los *clusters*, que se desarrollará en el Capítulo 3.

Sea w una cadena, i.e., $w \in \Sigma^*$ y a un carácter o símbolo. Denotamos con $|w|_a$ al número de veces que aparece el símbolo a en w. Es claro que,

$$|w| = \sum_{a \in \Sigma} |w|_a$$

Denotamos por $Pref(w)$ al conjunto de prefijos de w y $Suf(w)$ al conjunto de los sufijos de w. Para una pareja de enteros (i, j), tales que $0 \leq i \leq j \leq n-1$, notaremos mediante $w[i, j]$ a su factor $w[i, j] = w[i] \dots w[j]$.

Ejemplo: Para una cadena w =AGTCAGGA, los conjuntos $Pref(w) = \{A, AG, AGT, AGTC, AGTCA, AGTCAG, AGTCAGG\}$ y $Suf(w) = \{A, GA, GGA, AGGA, CAGGA, TCAGGA, GTCAGGA\}$, corresponden respectivamente a los prefijos y sufijos de w.

Además mencionamos algunas propiedades de las cadenas como:

- Una cadena v es un *reordenamiento* de u, si para toda $a \in \Sigma$

$$|u|_a = |v|_a$$

- Una cadena $u \in \Sigma^*$, es un factor finito de w si existen enteros $i, j \in \mathbb{N}$, $0 \leq i \leq j$, tales que $u = w_i \dots w_j$; la secuencia $w[i,j] = w_i \dots w_j$ la llamaremos una ocurrencia de u en w.

Ejemplo: Para w_1 =AGAGACT tenemos que el número de ocurrencias de AG en w_1 es 2.
Si $w_2 = 1112121212$ el factor 121, el número de ocurrencias en w_2 es 3.

Ahora bien, notemos con $F(w)$ al conjunto de todos los factores de la cadena w y con $Alph(w)$ al conjunto de caracteres del alfabeto Σ que ocurren en w.

Ejemplo: Para las cadenas $w = GCTCGAA$ y $u = ATTTTAAA$ tenemos que $Alph(w) = \{A, C, G, T\}$ y $Alph(u) = \{A, T\}$, respectivamente.

Un concepto muy importante para el desarrollo de esta investigación es el concepto de *complejidad de subcadenas* la cual definimos a continuación. Sea w una cadena, definimos a f_w la *complejidad de subcadenas de* w como la función $f_w : \mathbb{N} \longrightarrow \mathbb{N}$ tal que

$$f_w = card(F(w) \cap \Sigma^n).$$

Para todo n, $f_w(n)$ cuenta el número de factores distintos de longitud n que ocurren en w. Sea w una cadena finita, $f_w(n) = 0$ para $n > |w|$. Decimos que la cantidad

$$c(w) = card(F(w))$$

es la *complejidad total*, o *índice de complejidad* de w. Esta es una medida global de la riqueza de subcadenas que tiene una cadena w, y nos permitirá en el Capítulo 3 establecer un marco comparativo de nuestros objetos de estudio.

2.5. Factores especiales

Sea $card(\Sigma) = q$ y w una cadena de Σ^*. Para todo factor u de w consideremos el subconjunto maximal R_u de Σ, que denotaremos simplemente como R, tal que

$$uR \subseteq F(w).$$

Para todos las caracteres de $x \in R$ tenemos que $ux \in F(w)$, de lo contrario se tiene que para todo $x \in \Sigma \backslash R$, $ux \notin F(w)$, esto quiere decir que u ocurre en w seguido solo por cualquier carácter de R.

De manera simétrica, al considerar el subconjunto maximal L_u de Σ, que denotaremos L, tal que

$$Lu \subseteq F(w).$$

Para toda $y \in L$, se tiene que $yu \in F(w)$. Además, para todo $y \in \Sigma \setminus L$ tenemos que $y \notin F(w)$.

Ahora bien, introducimos la función $\mathfrak{v}_r : F(w) \longrightarrow \mathbb{N}$ definida para todo $u \in F(w)$ como

$$\mathfrak{v}_r(u) = card(uR) = card(R) = card(u\Sigma \cap F(w)).$$

A esta función la llamaremos *valencia a derecha de u*. De manera simétrica definimos la *valencia a izquierda de u* como

$$\mathfrak{v}_l = card(Lu) = card(L) = card(\Sigma u \cap F(w)).$$

Con el fin de analizar los factores a lado y lado de u, definimos la función $\mathfrak{v}_b(u)$ como la *valencia bilateral de u*, tal que $\mathfrak{v}_b : F(w) \longrightarrow \mathbb{N}$, definida para toda $u \in F(w)$ como

$$\mathfrak{v}_b = card(\Sigma u \Sigma \cap F(w)).$$

Decimos que un factor u es *especial a derecha* o *especial a izquierda*, cuando $\mathfrak{v}_r > 1$ o $\mathfrak{v}_l > 1$.

En consecuencia, un sufijo v de un factor especial a derecha u tiene valencia a derecha tal que $\mathfrak{v}_r(v) > \mathfrak{v}_r(u)$. De manera similar sucede con la valencia a izquierda de un prefijo de un factor especial a izquierda.

Ejemplo: Para $w = 1112121212$, la tabla 2.5 muestra algunos factores de w y sus respectivas valencias a izquierda y derecha:
Un factor se dice *biespecial* si es especial a derecha e izquierda. En consecuencia tenemos que si la valencia bilateral $\mathfrak{v}_b(u) > card(B)$ entonces u es *biespecial*.

Para cualquier j, tal que $0 \leq j \leq q$, denotamos por $S_r(j, w)$ al conjunto de todos los factores a derecha de w de valencia j, que nos permite definir:

$$s_r(j, n, w) = card(S_r(j, w) \cap \Sigma^n),$$

Factores	Valencia a izquierda	Valencia a derecha
1	2	2
2	1	1
12	2	1
21	1	1
121	2	1
212	1	1
1212	2	1
21212	1	1
121212	1	1

Cuadro 2.1: Valencias

para todo n. Note que $s_r(j, n, w)$ equivale al número de factores que tienen valencia a derecha igual a j. Más aún, establecemos que

$$g_r(j, n, w) = \sum_{k=j}^{q} s_r(k, n, w),$$

para cada n. Note que $g_r(j, n, w)$ es el número de factores de w con longitud n mayor e igual que j. Observamos que entre estas s_r y g_r se presenta una interesante relación:

$$\sum_{j=2}^{q}(j-1)s_r(j, n, w) = \sum_{j=2}^{q} g_r(j, n, w).$$

Las anteriores definiciones se presentan de manera simétrica para $S_l(j, w)$, y las funciones $s_l(j, n, w)$ y $g_l(j, n, w)$. Tenemos también que $g_r(2, n, w)$ ($g_l(2, n, w)$) iguales al número de factores especiales a derecha (a izquierda) de longitud n de w. En adelante, con el fin de simplificar la notación utilizaremos para $g_r(2, n, w)$, $R_w(n)$ y para $g_l(2, n, w)$ por $L_w(n)$.

Dado que los resultados que se obtengan a derecha son simétricos a izquierda, no habrá ambigüedad si notamos a $s_r(j, n, w)$ simplemente como $s(j, n)$.

La noción de factor especial es de gran importancia como lo indica [8], dado que la complejidad de subcadenas f_w de w, se da en términos de la enumeración de los factores especiales de w de diferentes valencias. En el capítulo final, observaremos algunos resultados relacionados con lo factores especiales y su relación con

la complejidad de unas cadenas especiales que se obtienen una vez se hacen los alineamientos múltiples de secuencias.

2.6. Complejidad de subcadenas

Para una cadena finita w de longitud m, encontramos consecuentemente una cantidad finita de factores que la conforman. Unos factores que son de especial interés en esta investigación son los sufijos, dado que ellos son los que definen los alineamientos de cadenas de RNAs a genomas de referencia, en la aplicación utilizada en este trabajo, ver 1.4.1. Por su definición, un sufijo es un factor que no tiene factores a derecha. Denotemos a u_0 como al sufijo minimal de w. Tenemos que cadena $\rho u_0 \in Suf(w)$, con $\rho \in F(w)$, no puede extenderse a derecha en $F(w)$.

Recordemos que $s_r(0, w)$ denota el número de todos los factores (sufijos) de la cadena w que tienen una valencia a derecha igual a 0, por lo que explicábamos anteriormente.

Si $|u_0| = k$, entonces tenemos que $s_r(0, n) = 0$ para $1 \leq n \leq k-1$ y $s_r(0, 1) = 1$ para $k \leq n \leq m$. Por lo tanto, el número total de factores de w que no se extienden a la derecha en w equivale a

$$\sum_{n=k}^{m} s_r(0, n) = m - k + 1.$$

Sea w una cadena de longitud finita y f_w la complejidad de subcadenas de w. Un resultado interesante, es la siguiente ecuación recursiva, para todo $n \geq 0$

$$f_w(n+1) = f_w(n) + \sum_{j=0}^{q} (j-1) s_r(j, n).$$

La anterior ecuación relaciona el número de factores de w de longitud $n+1$ con el número de factores de w de longitud n y la cantidad de factores especiales a derecha de longitud n, que tienen una valencia igual a 0.

En la iteración de la ecuación anterior, se obtiene la siguiente ecuación de complejidad de subpalabras de una cadena finita w:

$$f_w(n) = f_w(1) + \sum_{s=1}^{n-1} \sum_{j=0}^{q} (j-1) s_r(j, s).$$

Ahora, para todo $j \geq 2$, denotaremos con P_j el *número total de factores especiales a derecha de orden j.*

Proposición 2.1 (Prop. 4.1. de [8])**.**

$$\sum_{j\geq 2} P_j = m - k$$

Ahora nos referimos a los factores a izquierda de w. En este caso consideramos un prefijo v_0 de longitud minimal la cual no se extiende a izquierda en las subcadenas de w. Si tenemos a $h = |v_0|$. En una perfecta simetría se tiene que para todo $n \geq 0$, se tiene

$$f_w(n+1) = f_w(n) + \sum_{j=0}^{q}(j-1)s_l(j,n)$$

Para $j \geq 2$, sea Λ_j denota el *número total de factores especiales a izquierda de orden j.*

Proposición 2.2 (Prop. 4.2. de [8])**.**

$$\sum_{j\geq 2} \Lambda_j = m - h$$

De las proposiciones [2.1] y [2.2], obtenemos

$$\sum_{j=2}^{q}(P_j - \Lambda_j) = h - k.$$

Procederemos a estudiar la complejidad de subpalabras f_w de una cadena finita w sobre Σ de longitud m.

Observemos primero que una cota superior para la complejidad de subpalabras f_w, para todo $0 \leq n \leq m$,

$$f_w(n) \leq min\{q^n, m-n+1\}.$$

En efecto, q^n es la cardinalidad de Σ^n y $m-n+1$ es el número de ocurrencias de subcadenas de longitud n en w. Para la función $h : \mathbb{N} \longrightarrow \mathbb{N}$ definida para $n \geq 0$ se tiene

$$h(n) = min\{q^n, m-n+1\}$$

cuando n es lo suficientemente pequeño en relación con m, aumenta exponencialmente con n y decrece como una línea recta de pendiente -1, correspondiente al ángulo de $\frac{3\pi}{4}$.

Consideremos ahora, para cualquier n del número $R_w(n)$ de factores especiales de orden 2 y longitud n. Un sufijo de un factor especial a derecha es un factor especial a derecha; así que si existe un entero n tal que $R_w(n) = 0$, entonces $R_w(s) = 0$ para todo $s \geq n$. Dado que $R_w(m-1) = R_w(m) = 0$, de manera simplificada utilizaremos la notación R en lugar de R_w, definida como

$$R = min\{n | R_w(n) = 0\}.$$

Tenemos que $0 \leq R \leq m-1$. Si $R = 0$, la cadena w no tiene factores especiales a derecha para cualquier longitud $n \geq 0$. Esto ocurre si y sólo si w es una potencia de un solo caracter. Si $R > 0$, entonces $R-1$ representa la *longitud maximal de un factor especial en w*. Si $R = 1$, tenemos que en w no hay factores especiales a derecha de longitud $n \geq 1$. Este es el caso, por ejemplo la palabra $w = (ab)^k$ con $k > 0$.

Proposición 2.3 (Prop. 4.3. de [8]). *Sea la palabra w de longitud m tal que la $card(alph(w)) > 1$ y sea $s = min\{R, K\}$ y $S = max\{R, k\}$. La complejidad de subcadenas f_w es estrictamente creciente en el intervalo $[0, s]$, es no decreciente en el intervalo $[s, S]$ y estrictamente decreciente en el intervalo $[S, m]$. Más aún, para n en el intervalo $[S, m]$, se tiene que $f_w(n+1) = f_w(n) - 1$. Si $R < k$, entonces f_w es constante en el intervalo $[s, S]$.*

Proposición 2.4 (Prop. 4.4. de [8]). *La complejidad de subpalabras w toma su valor maximal en R y, más aun*

$$f_w(R) = \begin{cases} m - R + 1, & si \quad R \geq k; \\ m - k + 1, & si \quad R < k. \end{cases}$$

Si uno se refiere a factores a izquierda entonces tenemos equitativamente la cantidad

$$L = min\{n | L_w(n) = 0.\}$$

Proposición 2.5 (Prop. 4.5. de [8]). *La complejidad de subpalabras w toma su valor maximal en R y, más aun*

$$f_w(L) = \begin{cases} m - L + 1, & si \quad L \geq h; \\ m - h + 1, & si \quad L < h. \end{cases}$$

Corolario 2.1 (Coro. 4.1. de [8]). *La complejidad de subcadenas f_w es tal que $f_w(R) = f_w(L)$ y*

1. *$R = L$, si $R \geq k$ y $L \geq h$.*
2. *$R = h$, si $R \geq k$ y $L < h$.*
3. *$L = k$, si $R < k$ y $L \geq h$.*
4. *$h = k$, si $R < k$ y $L < h$.*

Proposición 2.6 (Prop. 4.6. de [8]). *Sea w una cadena y u un factor especial a derecha (izquierda) de w de longitud maximal. Si $R > h$ ($L > k$) entonces u es biespecial.*

Capítulo 3

Análisis de bloques de expresión mediante combinatoria de cadenas

3.1. Lenguajes

Las cadenas de *ARNs* se obtienen de la concatenación finita de los caracteres que representan a las moléculas Adenina(A), Citocina(C), Guanina(G) y Uracilo (U). Este conjunto de cadenas de *ARN*, al cual representaremos mediante el lenguaje $ARN^+ = \{A, G, C, U\}^+$, representa secuencias que en el contexto biológico cumplen con una amplia gama de funciones celulares.

Entre los posibles tipos de secuencias que hacen parte de ARN^+ encontramos a los *ARNts*, ver sección 1.3, para las cuales representaremos su lenguaje como $ARNt^+$. Si al alfabeto con que construimos este lenguaje le añadimos el carácter especial "_", que llamaremos *gap*, tendremos un nuevo lenguaje el cual notaremos como $\bar{ARNt}^+$. Las cadenas de este lenguaje resultan de aplicar un proceso de alineamiento o mapeo de cadenas, que han sido mencionadas en el Capítulo 1 como productos de secuenciamiento, a una cadena de tamaño mayor, o de referencia como puede ser un genoma de un organismo, y que corresponde a la manera de construir los bloques de ARNts o al realineamiento de lecturas de bloques de expresión previamente construidos en [4], cuyo contexto en combinatoria de palabras se describe a continuación.

A partir del lenguaje $\bar{ARNt}^+$ definimos un conjunto formado por cadenas cortas de longitud menor que 50. Es decir:

$$subARNt^{+} = \{w \in ARNt^{+} |\ |w| < 50\}.$$

Denotaremos $subARNt^{+}$ al lenguaje de las cadenas sometidas a un proceso de alineamiento o de mapeo a un genoma de referencia.

3.1.1. Relaciones entre cadenas

Dos cadenas t y z están relacionadas, si fueron alineadas por su similitud. A partir de esta relación definimos el conjunto $Sys = \{(t, z)|\ t, z \in ARNt^{+}\}$, el cual está formado por las parejas de cadenas de $ARNt^{+}$ que están relacionadas entre sí. En adelante, cuando una pareja de cadenas pertenezca a este conjunto diremos que está en un *formato común de coordenadas*, es decir que sus cadenas han sido alineadas de manera tal que se pueden realizar comparaciones entre caracteres en cada posición. La relación de similitud es una relación de equivalencia sobre el conjunto $ARNt^{+}$. Este conjunto define una relación de equivalencia.

De otra parte, notemos como $\Gamma_t \subseteq ARN^{+}$, con $t \in ARNt^{+}$, al conjunto de todos los α para los cuales existe un factor v de t, tal que α está alineado con v. En el conjunto Γ_t todas sus parejas están relacionadas entre sí, porque para cada cadena existe un factor en t al cual se alinean. A esta relación la notaremos por $\mathfrak{R}$.

Ejemplo: Las cadenas que se presentan a continuación corresponden a cadenas de ARNt originales, ver 3.1.4, a las cuales se les ha insertado en algunas posiciones gaps, con el propósito de ubicarlas en un formato en el cual se pueda comparar entre ellas los símbolos por posición.

Código ARNt	Cadena de caracteres
>559_hsa	GTTAAGA-T-G–GCAGAGCCCGGC—AATTGCA—T—A-A-AAC–T-TA-AAA
	-CTT-TATA————-A-CTGGAGGTTCAACTCCTC-T—TCTTCTTC
>127_hsa	GGCGCGG-T-G–GCCAAGT–GGT—-AAGGCG—T—C-G-GTC–T-CG-TAA
	-ACC-GAAG————-A-TCGCGGGTTCGAACCCCG-T—CCGTGCCT
>443_hsa	TCCCTGG-T-G–GTCTAGT–GGT—TAGGATT—C—G-G-CGC–T-CT-CAC
	-CGC-CGCG————-G–CCCGGGTTCGATTCCCG-G—TCAGGGAA

Cuadro 3.1: ARNts de una muestra de tejido de cerebro de humano

De manera que, en la posición 2 de cada una de ellas encontramos los caracteres T, G y C.

Sea $T = \{t_1, t_2, \ldots, t_n\}$ un conjunto de cadenas, todas ellas en formato único de coordenadas. El conjunto

$$\Gamma_T = \Gamma_{t_1} \cup \Gamma_{t_2} \cup \cdots \cup \Gamma_{t_n}$$

con Γ_{t_i}el conjunto de todas las cadenas α tal que $\alpha \mathfrak{R} t_i$, donde $1 \leq i \leq n$.

Tenemos entonces que a cada elemento de t le corresponde un conjunto Γ_t. Sea $\alpha \in \Gamma_t$, entonces existe un factor de t al cual está alineado. Ahora bien, si $t = t[0]t[1]\ldots t[m-1]$, entonces existen las posiciones i y f, con $0 \leq i < f \leq m-1$, que nos permiten expresar a t como la concatenación de factores

$$t = t[0 : i-1]t[i : f]t[f+1 : m-1],$$

se tiene $\alpha \mathfrak{R} t[i : f]$, es decir α se alinea al factor $t[i : f]$.

De esta manera, si tomamos los elementos pertenecientes a Γ_t podemos definir el conjunto de posiciones iniciales,

$$\mathbb{I}_{\Gamma_t} = \{i |\ 0 \leq i < f < m \text{ tales que existe } \alpha \mathfrak{R} t[i : f] \text{ con } \alpha \in \Gamma_t\}.$$

En ellas comienza el alineamiento entre los elementos de Γ_t y los factores de t.
De manera simétrica definimos el conjunto de las posiciones finales,

$$\mathbb{F}_{\Gamma_t} = \{f |\ 0 \leq i < f < m \text{ tales que existe } \alpha \mathfrak{R} t[i : f] \text{ con } \alpha \in \Gamma_t\},$$

Llamaremos *clusterset* al conjunto de todas las cadenas de Γ_t alineadas con algún factor de t, que inicia en la posición i y lo denotaremos por

$$Cl_i^t = \{\alpha \in \Gamma_t |\quad \alpha \mathfrak{R} t[i : f],\ 0 \leq i < f \leq m\},$$

Para este conjunto, definimos la siguiente relación de orden $\leq_{cl}$. Sean α, $\beta \in Cl_i^t$, decimos que $\alpha \leq_{cl} \beta$ si $\alpha \mathfrak{R} t[i : f]$ y $\beta \mathfrak{R} t[i : k]$, con $0 \leq i < f \leq k \leq m-1$, y $f \in \mathbb{F}_{\Gamma_t}$.

Gracias al orden $\leq_{cl}$ podemos definir en vector en el cual sus componentes están ordenadas así $\alpha_1 \leq_{cl} \alpha_2 \leq_{cl} \cdots \leq_{cl} \alpha_s$.

$$\overrightarrow{Cl_i^t} = \langle \alpha_1, \alpha_2, \ldots, \alpha_s \rangle,$$

A este vector lo llamaremos *cluster*.

3.1.2. Contorno del *cluster*

Otro objeto combinatorio que utilizaremos en este trabajo, es el *poliomino*, elemento ampliamente estudiado para definir teselaciones del plano. Aunque el estudio de sus propiedades no se ha presentado en detalle en este documento, en el capitulo final veremos como la caracterización de los contornos de los poliominos mediante la concatenación de caracteres, generando cadenas, nos dará una opción de analizar grandes cantidades de información de alineamientos múltiples.

A continuación presentamos algunas definiciones.

Sea R una región del plano cartesiano, decimos que una teselación $\mathbb{T}$ es un cubrimiento mediante regiones más pequeñas que no se superponen, ni dejan huecos o espacios entre ellas. En el caso en que las regiones sean cuadrados unitarios, los llamaremos celdas. Ahora bien, dada una región R y $\mathbb{T}$ una teselación mediante celdas, se define un poliomino $\mathbb{P}$ como un conjunto de celdas que conforman una teselación y comparten solo un lado

En la figura 3.1.2, se muestran poliominos conformados por dos, tres y cuatro celdas.
Ejemplo: Dominos, triominos, tetraminos.

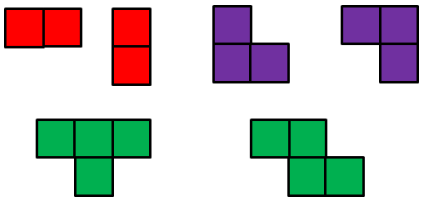


Figura 3.1: Conformados por dos, tres y cuatro celdas

Dado un poliomino $\mathbb{P}$, este se dice

- *Vertical:* Si la intersección entre cualquier recta vertical que pase por la región que cubre $\mathbb{P}$ es un segmento vertical.
- *Horizontal:* Si la intersección entre cualquier recta horizontal que pase por la región que cubre $\mathbb{P}$ es un segmento horizontal.
- *Convexo:* Si es un poliomino vertical y horizontal.

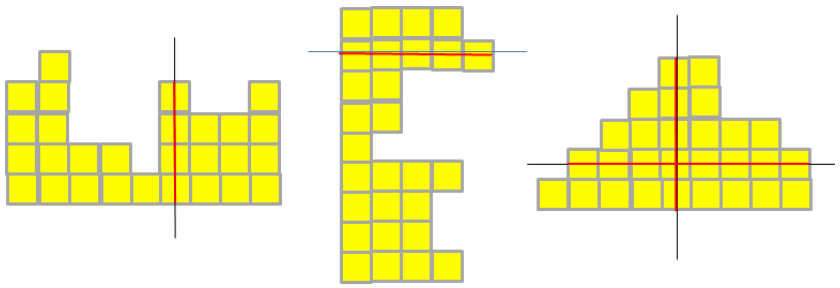


Figura 3.2: El primer poliomino a la izquierda corresponde a uno vertical, el del medio corresponde a un poliomino horizontal y el de la derecha corresponde a uno convexo

Si se establecen una serie de movimientos en los lados de las celdas, ver [32] y si se codifican convenientemente así:

$$0 = (1, 0), 1 = (0, 1), 2 = (-1, 0), 3 = (0, -1),$$

asignando a cada movimiento un carácter, se tiene que $\Sigma = \{0, 1, 2, 3\}$ es el alfabeto que codifica los cuatro movimientos posibles en los contornos de una celda. Entonces de la concatenación ellos resultan cadenas, que representan un camino a lo largo de los lados de las celdas. Esta relación entre objetos discretos y cadenas se ha utilizado en el modelado de los problemas de las teselaciones en el plano con poliominos, ver [3] [28] y [25], permitiendo así sintetizar la representación.

Ejemplo: A continuación presentamos un poliomino conformado por las celdas que se resaltan en color amarillo,

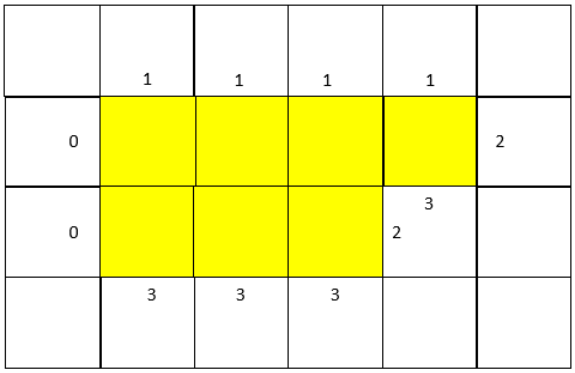


Figura 3.3: Si iniciamos un recorrido por los lados externos del poliomino, iniciando en la esquina inferior izquierda, la cadena $w = 001111232333$ representa los movimientos que realizamos para recorrer el contorno del mismo.

El uso de cadenas de caracteres para el estudio de los contornos de los poliominos, es una técnica muy utilizada y efectiva para su modelación. Es así, que haciendo uso de una representación similar, una vez tenemos $\overrightarrow{Cl_i^t}$, representaremos una ruta en su contorno, codificando los siguientes movimientos asi:

- $1 :=\rightarrow$, de izquierda a derecha.
- $2 :=\uparrow$, de abajo a arriba

.
Estos movimientos, nos permitirán identificar el contorno o ruta que describen cadenas alineadas, si con ellas formaramos un poliomino, en el cual cada celda corresponde a un carácter del cluster. Este contorno caracteriza los movimientos que se hacen para recorrer de izquierda a derecha las posiciones que cuentan con alineamientos y de abajo a arriba la cantidad cadenas que forman el cluster.
Finalmente, representaremos el *contorno de* Cl_i^t con la cadena

$$\mathcal{P}_i^t = 1^{|\alpha_1|} \cdot 2 \cdot 1^{|\alpha_2|-|\alpha_1|} \cdot 2 \cdot 1^{|\alpha_3|-|\alpha_2|} \cdot 2 \cdots 1^{|\alpha_s|-|\alpha_{s-1}|} \cdot 2,$$

la cual tiene tantos *1s* como posiciones de t se alinean con las cadenas de Cl_i^t y tantos $2s$ como cadenas pertenecen a Cl_i^t.

Ejemplo: La cadena t que se ve en miniatura, representa un ARNt de la base de datos objeto de nuestro estudio la cual contiene una subcadena que resaltamos en color rojo. Como se observa en la Figura 3.4 hay un alineamiento de las subcadenas $\alpha_1, \alpha_2 \ldots \alpha_{12}$,

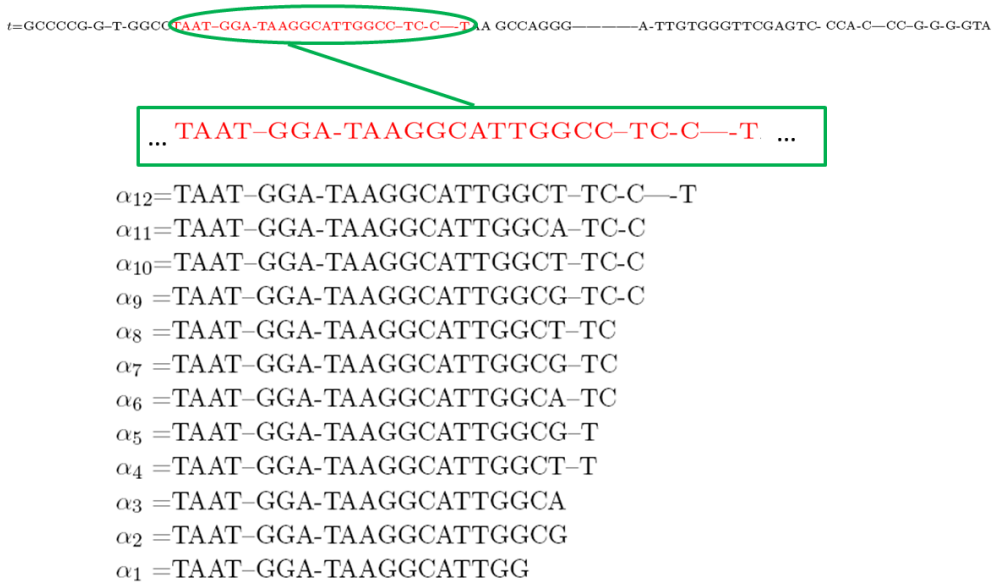


Figura 3.4: Generado desde su posición 16 a partir de las subcadenas alineadas $\alpha_1, \alpha_2 \ldots \alpha_{12}$

Las cadenas que se presentan en la Figura 3.4, una vez alineadas, han sido ordenadas por su longitud, conformando el vector $\overrightarrow{Cl}^t_{16}$, es decir el cluster de la cadena t en la posición 16.

A partir del cluster, la cadena $\mathcal{P}^t_{16}$ que representa el contorno del cluster es la siguiente,

$\mathcal{P}^t_{16} = 1^{20} \cdot 2 \cdot 1^{(23-21)} \cdot 2 \cdot 1^{(23-23)} \cdot 2 \cdot 1^{(25-23)} \cdot 2 \cdot 1^{(25-25)} \cdot 2 \cdot 1^{(26-25)} \cdot 2 \cdot 1^{(26-26)} \cdot 2 \cdot 1^{(26-26)} \cdot 2 \cdot 1^{(26-26)} \cdot 2 \cdot 1^{(28-26)} \cdot 2 \cdot 1^{(27-27)} \cdot 2 \cdot 1^{(27-27)} \cdot 2 \cdot 1^{(27-27)} \cdot 2 \cdot 1^{(31-27)} \cdot 2,$

simplificando se obtiene la cadena

$\mathcal{P}^t_{16} = 11111111111111111111211221122122211222111112$

Nota: Para una cadena de ARNt, es posible que le corresponda una gran cantidad de cadenas alineadas. En la Figura 3.5 presentamos 261 cadenas que han sido alineadas a un ARNt y que llamaremos w. En el eje horizontal tenemos las posiciones en la cadena w y en el eje vertical las cadenas que le son alineadas. Con color azul rey representamos los lugares en los que no hay alineamientos o que corresponden gaps y con los demás colores representamos los lugares que están alineados.

Como se observa, mediante el procedimiento descrito, y como se puede ver en el ejemplo 3.1.2, estas 261 cadenas se pueden reorganizar de tal manera que se agrupan por la posición de inicio y longitud. Este procedimiento permite simplificar un cluster a una nueva representación que corresponde a la cadena que describe su contorno.

De otra parte, si tenemos una cadena $\mathcal{P}_i^t$ correspondiente a un cluster $\overrightarrow{Cl_i^t}$, conformado por las cadenas $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_s$, la longitud de $\mathcal{P}_i^t$ está acotada por

$$|\mathcal{P}_i^t| \leq \binom{n + |\alpha_s| - |\alpha_1| - 2}{|\alpha_s| - |\alpha_1|}.$$

Esto se tiene ya que los posibles pasos que se pueden dar en el contorno que hemos definido, inician en el final de α_1 y llegan al final α_s de $\overrightarrow{Cl_i^t}$. Por lo tanto, la cantidad de movimientos a derecha equivalen a cantidad de caracteres que hay entre la cadena más larga del cluster α_s y la más corta α_1, es decir $|\alpha_s| - |\alpha_1|$. En relación con los movimientos hacía arriba, la máxima cantidad es $n - 2$, dado que se deben descontar 2 movimientos, el que corresponde al final de la cadena $|\alpha_1|$ y el del final de $|\alpha_s|$. Teniendo así los movimientos en las dos direcciones que conectan α_1 con α_s. Debemos seleccionar de los $n + |\alpha_s| - |\alpha_1| - 2$ los correspondiente a derecha, es equivalente si seleccionamos los que van hacia arriba.

3.1.3. Datos y procesamiento

Las cadenas que fueron objeto de este análisis corresponden a ARNs pequeños aislados de tejido de la corteza prefrontal congelada de un cerebro humano. Un total de 19522 lecturas fueron mapeadas a 421 ARNts del genoma humano. Para obtener el conjunto de lecturas de entrada para construir los bloques fue necesario secuenciar los ARNs utilizando tecnología HTS de tipo *Small RNA Preparation Protocol (Illumina, USA)*. Una vez obtenidas las lecturas, éstas fueron alineadas al genoma humano (NCBI36.50 Publicación de julio de 2008) utilizando el software Segemehl, ver [14]. Recordemos que Segemehl que utiliza un método basado en una variación de las matrices mejoradas de sufijos, y estadísticas de coincidencia que evalúa el grado de error de los alineamientos en cuanto a sustituciones, inserciones y borramientos en las lecturas producidas por el error de la técnica de secuenciamiento. Una vez mapeadas las secuencias se construyeron bloques de expresión, ver [4], para detectar patrones de expresión específicos que en nuestro caso simplemente los llamaremos bloques, ver [21, 20, 4]. Del conjunto de bloques obtenidos de todo el genoma humano, se seleccionaron para fines de este trabajo los correspondientes a los sobrelapados con las regiones del genoma en las que se ubican los genes de

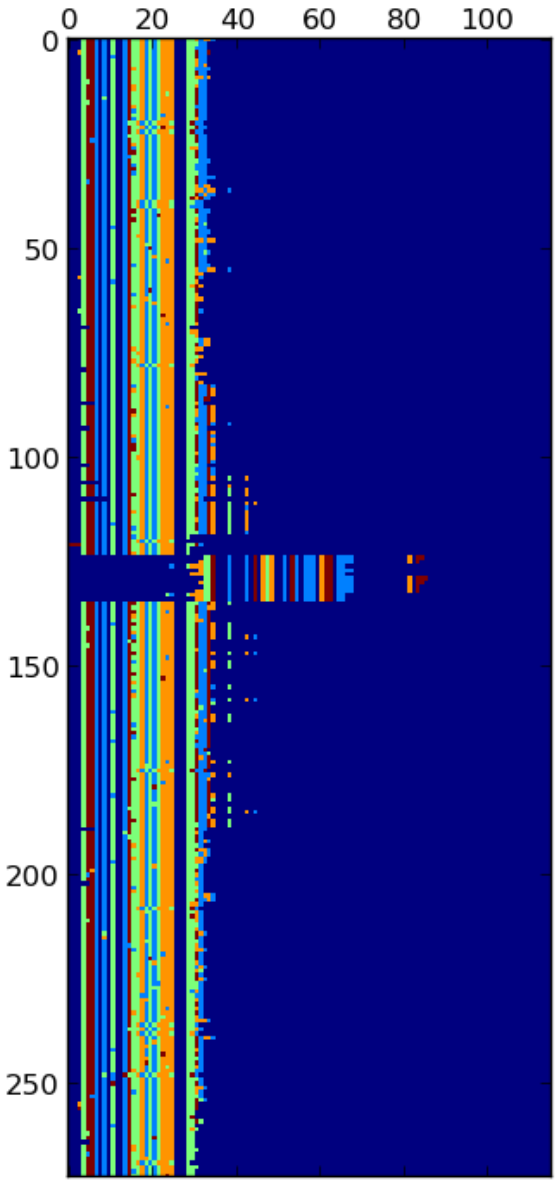


Figura 3.5: 261 cadenas han sido alineadas a un ARNt. En el eje horizontal tenemos las posiciones en de la cadena del ARNt y en el eje vertical las cadenas que le son alineadas. En color azul rey representamos los lugares en los que no hay alineamientos o que corresponden gaps y con los demás colores se representan los lugares que están alineados al ARNt.

los ARNts.

Aunque la información que se tiene tanto de los ARNts, como de sus copias, ya que es posible encontrar varias regiones que codifican para un mismo tipo ARNt, comprende otras características relevantes en el contexto bioinformático, en este trabajo se usan para poder tipificar sus contornos. Esta metodología se puede aplicar y generalizar para cualquier bloque de expresión construido bajo la metodología referida.

De los bloques construidos asociados a los 421 ARNts realizamos el procesamiento de sus cadenas constitutivas basados en los elementos teóricos que han sido presentados en el Capítulo 2. Basados en ellos, se procedió al diseño de los pseudocódigos requeridos para contextualizar el problema teórico con su posterior implementación en lenguaje de programación Python. Los seudocódigos que presentamos a continuación fueron desarrollados e implementados para reconstruir los contornos asociados a los bloques de expresión en ARNts dentro del contexto de esta tesis.

3.1.4. Datos de entrada para los programas (Inputs)

En la preparación de los datos que permitieron estructurar los archivos iniciales, se utilizó el programa *Segemehl*[14] y *Blockbuster*.

Los siguientes cuadros corresponden a la información que utilizamos tanto de los ARNts como de sus copias en el genoma humano. Dado que nuestro interés es construir los clusters de cadenas y solo trabajaremos con las variables que sobre ellos definimos.

Información de los ARNts

- t : Cadena de caracteres (A, C, G, T, _) que conforma a t en *formato común de coordenadas*, es decir $t \in ARNt\_$
- $Grupo\_codon$: Grupo codón de t.
- Cod_t : Código del t (etiqueta para diferenciar los ARNts).

Ahora para cada copia α (estas no tienen gaps dado que aún no están en formato común de coordenadas y no han sido alineadas), tenemos la siguiente información:

Información de las copias

- α : Cadena de caracteres (A, C, G, T) que conforma a la cadena α (sin

gaps).

- t : Cadena de $ARNt\_$ a la cual está alineada α.
- Cod_α : Código de α (etiqueta que permite diferenciarla).
- i : Posición inicial de la subcadena de t a la cual está alineada α (previo a ubicación en formato común de coordenadas).
- f : Posición final de la subcadena de t a la cual está alineada α (antes de que se la ubicaran los ARNts en formato común de coordenadas).

3.1.5. Seudocódigos

A continuación presentamos algunos de los seudocódigos que representan la abstracción de los pasos requeridos para el procesamiento realizado para descubrir patrones sobre la complejidad de los contornos de los bloques de ARNts. Inicialmente presentamos la información de los archivos que surgieron de los procesos explicados en la sección 3.1.4.

El seudocódigo que presentamos a continuación corresponde al proceso previo a la construcción de los clusters. Dado que los ARNts, previamente se les ha insertado gaps con el fin de ubicarlos de manera que sus caracteres en cada posición sean comparables. Con este procedimiento se logra insertar los gaps a cada una de las copias que tiene asociadas determinado ARNt, con el propósito de tenerlos en un formato común de coordenadas.

Sea T el conjunto de los $ARNt\_$, conformado por $t_1, t_2, \ldots, t_n$ y Γ_t el conjunto de los $\alpha_1, \alpha_2, \ldots, \alpha_s$ que se alinean a t:

Algorithm 1 Formato común de coordenadas a elementos de Γ_t

```
1: function FCC( Γ_t)
2:     longitud_t ← longitud de la cadena t
3:     PosiGaps = []                                   ▷ Lista vacia
4:     for Cada 0 ≤ j < longitud_t do          ▷ Posiciones de la subcadena
5:         if t[j] = "_" then                          ▷ Identificación de gaps
6:             PosiGaps+ = j                       ▷ Agregar j a la lista
7:     for Cada α de Γ_t do                                ▷ Inserta gaps
8:         longitud_α ← longitud de la cadena α
9:         for Cada j de PosiGaps do
10:            if i ≤ j then
11:                i ← i + 1
12:            if i ≤ j then
13:                s ← j − i
14:                α ← α[0 : s − 1] + "_" + α[s + 1 : longitud_α − 1]
15:                f ← f + 1
       return Γ_t                          ▷ Γ_t en formato común de coordenadas
```

Una vez se han insertado los gaps a las cadenas de Γ_t, se debe realizar el realineamiento de ellas a t dado que han sufrido modificaciones. Para ello, como se indicaba al inicio de la sección, utilizamos el programa *ClustalW*[1] acoplado a *BioPython*[2], a partir del cual se obtienen los realineamientos de secuencias y sus puntuaciones respectivas, las cuales corresponden a porcentajes de similitud entre cadenas.

De otra parte, la longitud de cada cadena de Γ_t una vez se le inserta un gaps, incrementa en una unidad. Como se observa en el algoritmo anterior cada vez que encuentra un gap en la cadena t, se inserta un gap en cada subcadena α, en la misma posición. Lo anterior en cuanto a que con la información obtenida con *Segemehl*, ver 1.4.1, hace falta que estas cadenas queden en *formato común de coordenadas*, es decir se puedan comparar sus caracteres por posición.

Una vez se insertan los gaps a cada subcadena, es necesario volver a realinearlas a la cadena t, este proceso se realiza mediante el algoritmo 0, que se presenta a continuación:

[1] Ver http://www.clustal.org/

[2] ver www.biopython.org.

Algorithm 2 Realineamiento de los elementos de Γ_t

function REALIGN(Γ_t,$\mathbb{I}_{\Gamma_t}$,$\mathbb{F}_{\Gamma_t}$) ▷ Cada $\alpha \in \Gamma_t$, con gaps, ver alg.[0]
▷ Con las posiciones iniciales y finales formamos $\mathbb{I}_{\Gamma_t}$ y $\mathbb{F}_{\Gamma_t}$

Sean Γ_t, $\mathbb{I}_{\Gamma_t}$ y $\mathbb{F}_{\Gamma_t}$ ▷ Listas que guardan correspondencia entre sí
$Min\_\mathbb{I}_{\Gamma_t} \leftarrow$ Mínimo de $\mathbb{I}_{\Gamma_t}$
$Max\_\mathbb{F}_{\Gamma_t} \leftarrow$ Máximo de $\mathbb{F}_{\Gamma_t}$

for Cada α de Γ_t **do**
Alinear α a $t[Min\_\mathbb{I}_{\Gamma_t} : Max\_\mathbb{F}_{\Gamma_t}]$ ▷ Con ClustalW
Sea $Aligns\_\alpha$ ▷ Lista de alineamiento obtenidos
Sea $Puntaje\_\alpha$ ▷ Lista de sus puntuaciones respectivas
$Max\_Puntaje\_\alpha$ ▷ Mejor puntaje de los alineamientos
$Best\_Aligns = []$ ▷ Lista vacia, mejores alineamientos
$long\_Puntaje\_\alpha \leftarrow$ longitud de $Puntaje\_\alpha$
for Cada $0 \leq k \leq long\_Puntaje\_\alpha - 1$ **do**
if $p = Max\_Puntaje\_\alpha$ **then**
Best_Aligns+=Aligns_$\alpha[p]$ ▷ Agrega a la lista
if $long\_Puntaje\_\alpha > 0$ **then**
$\alpha = Aligns\_\alpha[1]$ ▷ Cambia α por alineamiento
$long\_\alpha \leftarrow$ Longitud de α ▷ Recalcula longitud de α
$f \leftarrow i + long\_\alpha$
return Γ_t

Al utilizar los algoritmos anteriores hemos logrado, redefinir los elementos de Γ_t, haciendo que estén en formato común de coordenadas y luego realineados a t.

Ahora bien, procederemos a mostrar cómo se construyen los clusters Cl_i^t, ver 3.1.1. Cabe anotar que con el algoritmo anterior las posiciones a las que queda alineada la cadena α cambian, por eso se actualiza f.

Con el seudocódigo que presentamos a continuación ordenamos las copias asociadas a los ARNts, ordenando mediante el algoritmo de burbuja, primero por las posiciones iniciales y luego por las finales a las cuales son alineadas. De manera que obtenemos como resultado los clusters por posición, ver la descripción del algoritmo 0.

Algorithm 3 Construcción de clusters $\overrightarrow{Cl_i^t}$

function CLUSTERS(Γ_t, $\mathbb{I}_{\Gamma_t}$,$\mathbb{F}_{\Gamma_t}$) ▷ Los inputs son listas
  $m \leftarrow$ longitud de la lista Γ_t ▷ $\mathbb{I}_{\Gamma_t}$,$\mathbb{F}_{\Gamma_t}$ tienen el mismo tamaño
  Sean x, y, z variables auxiliares
  for Cada $0 \leq j < m$ **do** ▷ Implementación del algoritmo de burbuja
    for Cada $1 \leq k \leq m - j$ **do**
      if Si $\mathbb{I}_t[j] > \mathbb{I}_t[j+1]$ **then**
        $x \leftarrow \Gamma_t[j]$ ▷ Ordenando Γ_t
        $\Gamma_t[j] \leftarrow \Gamma_t[j+1]$
        $\Gamma_t[j+1] \leftarrow x$
        y=$\mathbb{I}_t[j]$ ▷ Ordenando $\mathbb{I}_t$
        $\mathbb{I}_t[j] \leftarrow \mathbb{I}_t[j+1]$
        $\mathbb{I}_t[j+1] \leftarrow y$
        $z \leftarrow \mathbb{F}_t[j]$ ▷ Ordenando $\mathbb{I}_t$
        $\mathbb{F}_t[j] \leftarrow \mathbb{F}_t[j+1]$
        $\mathbb{F}_t[j+1] \leftarrow z$
        if $\mathbb{F}_t[j] > \mathbb{I}_t[j+1]$ **then**
          $x \leftarrow \Gamma_t[j]$ ▷ Ordenando Γ_t
          $\Gamma_t[j] \leftarrow \Gamma_t[j+1]$
          $\Gamma_t[j+1] \leftarrow x$
          $y \leftarrow \mathbb{I}_t[j]$ ▷ Ordenando $\mathbb{I}_t$
          $\mathbb{I}_t[j] \leftarrow \mathbb{I}_t[j+1]$
          $\mathbb{I}_t[j+1] \leftarrow y$
          $z \leftarrow \mathbb{F}_t[j]$ ▷ Ordenando $\mathbb{F}_t$
          $\mathbb{F}_t[j] \leftarrow \mathbb{F}_t[j+1]$
          $\mathbb{F}_t[j+1] \leftarrow z$
  $Min\_\mathbb{I}_{\Gamma_t} \leftarrow$ Mínimo de $\mathbb{I}_{\Gamma_t}$
  $Max\_\mathbb{F}_{\Gamma_t} \leftarrow$ Máximo de $\mathbb{F}_{\Gamma_t}$
  for Cada $Min\_\mathbb{I}_{\Gamma_t} \leq i < m - 1$ **do**
    $\overrightarrow{Cl_i^t} = []$ ▷ Lista vacía
    for Cada $0 \leq k < |\Gamma_t|$ **do**
      if $i = |\Gamma_t[k]|$ **then**
        $\overrightarrow{Cl_i^t} += \Gamma_t[k]$
    return $\overrightarrow{Cl_i^t}$
  return 0

Dado que tenemos como resultado del algoritmo 0 obtenemos cada uno de los

clusters asociados por posición al ARNt respectivo. Con el seudocódigo siguiente construimos la cadena que representa al contorno del cluster, contorno que inicia en la primera posición a la que está alineada la primera copia del cluster, siguiendo los movimientos, como se muestra en el ejemplo ver 3.1.2, hasta llegar a la posición a donde llega la última copia,.

Algorithm 4 Construcción de la cadena de contorno $\mathcal{P}_i^t$

function CONTOUR($\overrightarrow{Cl_i^t}$)
  $\overrightarrow{l_i^t} = []$ ▷ Lista vacia
  for Cada α de $\overrightarrow{Cl_i^t}$ **do**
    $long\_\alpha \leftarrow$ longitud de α
    $\overrightarrow{l_i^t} += long\_\alpha$
  $long\_\overrightarrow{l_i^t} \leftarrow$ longitud de $\overrightarrow{l_i^t}$
  $PrefContour \leftarrow 1^{|}\alpha_1| \cdot 2$ ▷ Lista vacia, repr. movimientos → y ↑
  for Cada $2 \leq k \leq long\_\overrightarrow{l_i^t}$ **do**
    $Factor = 1^{|\alpha_k|-|\alpha_{k-1}|} \cdot 2$
  $\mathcal{P}_i^t = PrefContour + Factor$ ▷ Concatenación de cadenas
  return $\mathcal{P}_i^t$

La búsqueda de los factores que componen las cadenas de contorno la podemos realizar de distintas maneras. Una de ellas es generando un factor determinado y verificando si corresponde a una subcadena de la cadena de contorno, esta verificación es la que se realiza con el seudocódigo 5. El cual retorna *verdadero* si es una subcadena y *falso* en caso contrario.

Algorithm 5 Subcadena de w

```
function SUBCADENA( u, w)
    m ← longitud de u
    n ← longitud de  w
    (i, j) ← (0, 0)
    for Cada 0 < i < m y cada 0 < j < n do
        if  thenv[i] = w[j]
            i ← i + 1
        j ← j + 1
    if i = m then
        return Verdadero
    if i ≠ m then
        return Falso
    return 0
```

Otra forma de determinar la complejidad es recorriendo la palabra de contorno con dos índices, entre los cuales se encuentra el factor. Esta alternativa resulta mucho más eficiente que la planteada para el seudocódigo 5.

Luego de tener los factores, se va añadiendo a una lista y se eliminan los factores repetidos. La cantidad de factores encontrados corresponde a la complejidad de subcadenas para cada cadena de contorno.

Algorithm 6 Complejidad de subcadenas de $\mathcal{P}_i^t$

```
function COMPLEX(P_i^t)
    n ← longitud de P_i^t
    Sub=[]                          ▷ Lista vacía, subcadenas de w
    for Cada 0 < i < n do
        for Cada i < j <n do
            factor=P_i^t[i : j]
            Subs+=factor            ▷ Agrega a la lista subcadenas
    procedure E(l)iminar elementos repetidos de Subs
        c ← tamaño de Subs return c     ▷ Complejidad de subcadenas
```

3.2. Resultados

Las figuras que se presentan en esta sección son el resultado de la aplicación de los algoritmos presentados anteriormente, implementados para el procesamiento de una librería que contiene la información de ARNts y sus copias, ver 3.1.4. Esta información corresponde aproximadamente a un millardo de caracteres concatenados en cadenas. Los datos fueron procesados inicialmente con *Segemehl* y *Blockbuster*. Los códigos diseñados fueron programados en el lenguaje de programación Python, y en particular se utilizó el programa *ClustalW* de las librería *Biopython*, para el realineamiento de cadenas. En cuanto a las gráficas, estas se obtuvieron mediante el uso de las librerías *numpy* y *mathplotlib*.

3.2.1. Repeticiones de subcadenas en las cadenas de contornos

La complejidad total de las cadenas corresponde a la cantidad de subcadenas que en ella se encuentran. Un elemento fundamental para la caracterización de la complejidad, corresponde a la existencia de factores y la cantidad de veces que aparecen en cadenas de contornos.

En las figuras 3.6 y 3.7 observamos que para una cadena de contorno w (la cual se encuentra en la parte superior del gráfico), en el eje horizontal se presenta el índice de cada subcadena (en orden lexicográfico y numeradas de menor a mayor), y en el eje vertical la cantidad de veces que aparece en la cadena de contorno.

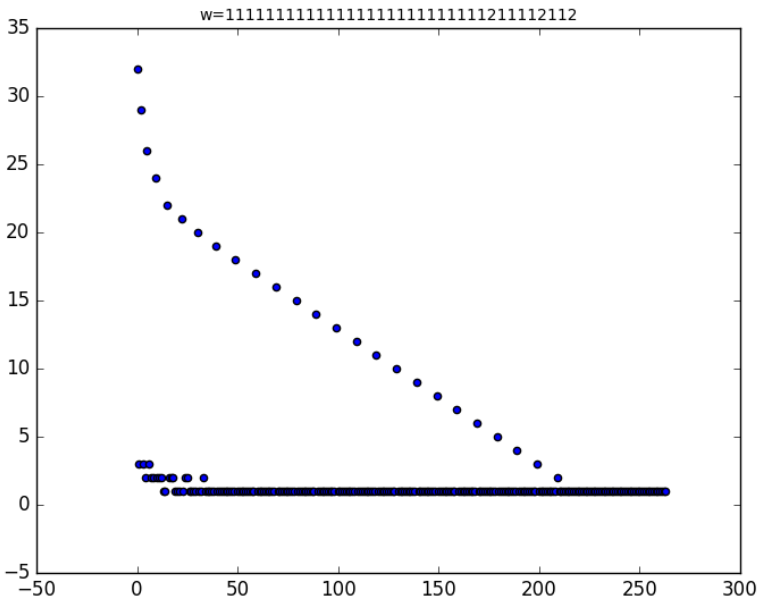


Figura 3.6: Esta cadena es w que aparece en encabezado de la gráfica. En el eje horizontal se representan el índice de los factores ordenados por tamaño, iniciando con el primer factor *1*, el segundo *2*, seguido de *11* y luego *12*, sucesivamente hasta llegar a la misma w. En el eje vertical se muestra la cantidad de veces que cada factor aparece en la cadena. Por ejemplo, el factor *1* aparece 32 veces, *2* lo hace 3 veces, *11* se encuentra 31 veces y así sucesivamente.

En la figura 3.6, se muestra que existen muchos factores que aparecen solamente una vez, representados por puntos que se forman en la paralela al eje horizontal. La Figura 3.6 muestra que los factores, para que no se forman en una línea paralela al eje horizontal, corresponden a factores especiales a izquierda o a derecha, es decir que tienen valencia superior a 1. Por ejemplo, las subcadenas *1*, *2* y *12*, tienen valencia mayor de 1.

De la misma manera, en la Figura 3.7 se presenta otra cadena de contorno para la cual se realizó el mismo procedimiento.

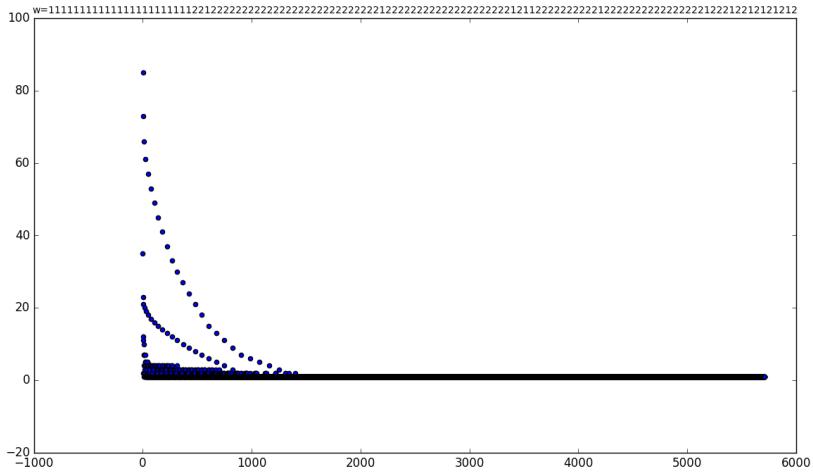


Figura 3.7: Ocurrencias de factores en una cadena donde predomina el caracter *2*.

En este caso se observan dos tendencias en las repeticiones de factores, conformadas por los factores donde aparece el caracter 2 y otras por el 1.

En ella observamos que los puntos que representan la cantidad de veces que aparecen factores, forman dos curvas de tendencia, correspondientes a las repeticiones de factores conformadas por el carácter *2*, como la de mayor inclinación, y otra por las formadas por el factor *1*. De esta manera se evidencia que las valencias de estos factores de la primera curva superan a la segunda.

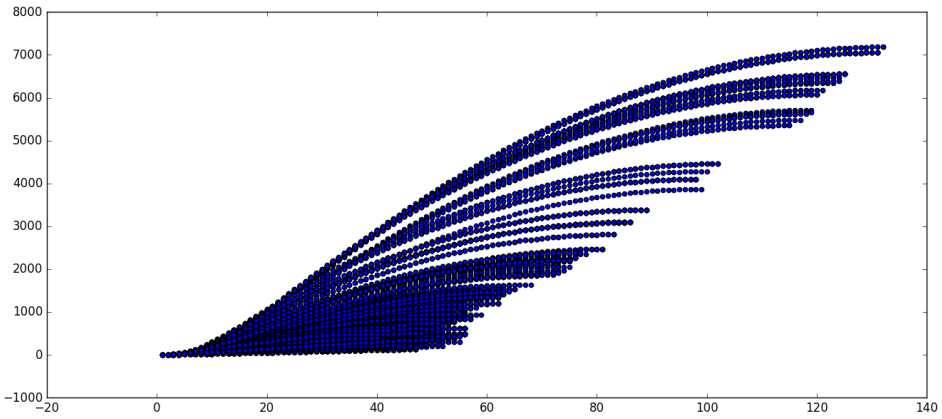


Figura 3.8: En el eje horizontal se encuentra el índice de factores ordenados y en el vertical la cantidad acumulada de veces que se repite en la cadena.

La Figura 3.8, muestra la densidad de factores en todas las cadenas de contorno encontradas para un ARNt particular. En ella podemos observar el comportamiento acumulado de subcadenas, sin contar las ocurrencias de cada una, de manera que en el eje horizontal se encuentra el índice de las subcadenas ordenadas y en el vertical la cantidad acumulada encontrada.

3.2.2. Complejidad total por grupos de codones

La complejidad total de una cadena de caracteres permite a partir de la cantidad de subcadenas, establecer un marco de comparación para caracterizar los contornos de los clusters objeto de estudio. De igual manera nos permite caracterizar las variaciones de las longitudes de las copias que se alinean iniciando en una posición determinada, ver ejemplo de la sección 3.1.2.

Las gráficas que presentamos a continuación corresponden al cálculo de la complejidad de subpalabras para cada una de las cadenas de contornos, calculada a su vez para cada uno de los clusters encontrados. Cada gráfica, muestra para todos los ARNts pertenecientes al mismo grupo codón, en el eje horizontal la posición donde inicia el contorno y en el eje vertical los valores respectivos de complejidad de la cadena del contorno. De esta manera cualquiera de ellas reúne a todas las gráficas de las cadenas del mismo grupo.

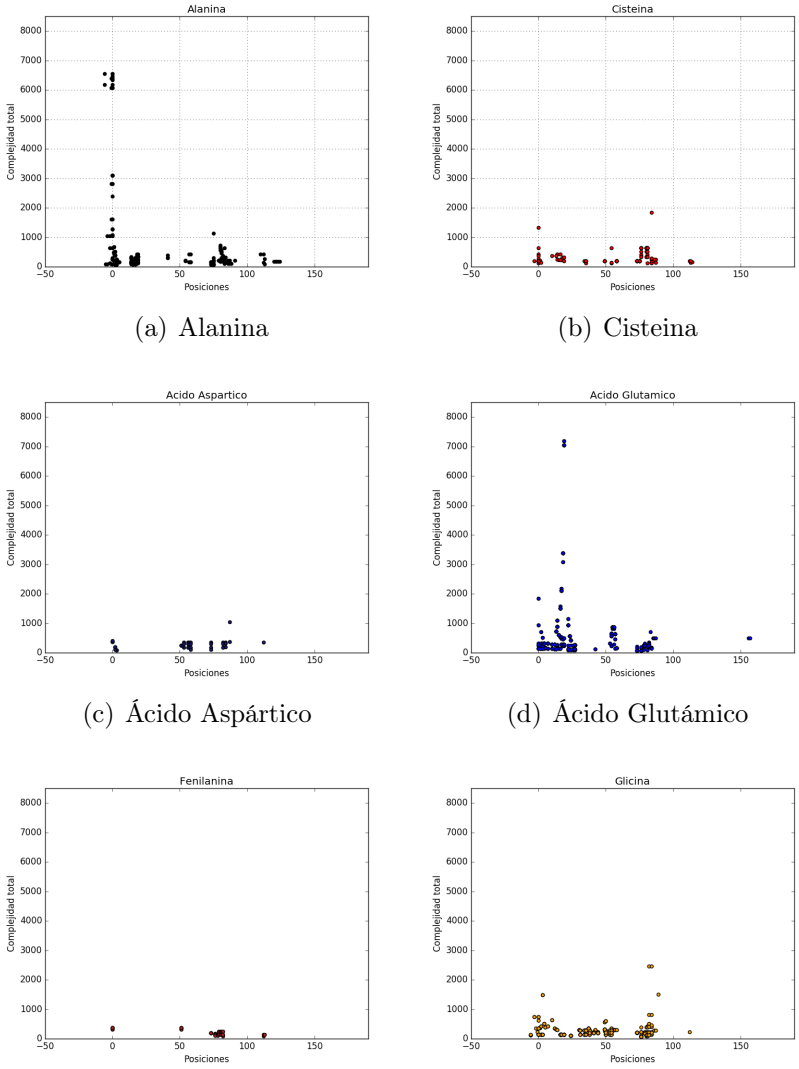
Por ejemplo, para el grupo codón Alanina se buscan todos los ARNts del grupo (información que se especifica en los archivos iniciales de los datos), después de aplicar los algoritmos, se tiene para cada ARNt de Alanina todos los cluster por posición de alineamiento. Con cada Cluster al aplicar el algoritmo 4, se obtiene la cadena que representa al contorno, sobre las cuales finalmente se calcula la complejidad total y se genera la respectiva gráfica de *posición* versus *complejidad*. La composición de todas las gráficas obtenidas de los ARNts de Alanina, es la que presentamos bajo el mismo nombre.

Estos grupos se forman de acuerdo con el aminoácido que facilita el proceso de transcripción del ARNt. La tabla que se presenta a continuación, especifica los nombres de los distintos grupos codones analizados y sus respectivas siglas.

Grupo	Aminoácido	Abreviatura
1	Alanina	Ala
2	Cisteina	Cys
3	Ácido espártico	Asp
4	Fenilanina	Phe
5	Glicina	Gly
6	Histidina	His
7	Isoleucina	Ile
8	Lisina	Lys
9	Leuicina	Leu
10	Metionina	Met
11	Asparagina	Asn
12	Prolina	Pro
14	Glutamina	Gln
15	Arginina	Arg
16	Serina	Ser
17	Treonina	Thr
18	Valina	Val
19	Triptofano	Trp
20	Tirosina	Tyr

En las Figuras 3.9, 3.10, 3.11 y 3.12, se muestra para cada uno de los grupos codones, en las posiciones donde se registran alineamientos, la complejidad total de su cluster asociado. Es así, que hay posiciones en las que se presentan múltiples valores de complejidad, lo que indica que entre toda la muestra de ARNts, las cadenas del mismo grupo tienen cada una, un cluster para el que su cadena de

contorno tiene una complejidad que representamos con un punto. En la Figura 3.13, encontramos los valores de complejidad total para todas las palabras de contorno de los clusters registrados, diferenciando con colores cada uno de los grupos de codones utilizados en las Figuras 3.9, 3.10, 3.11 y 3.12.

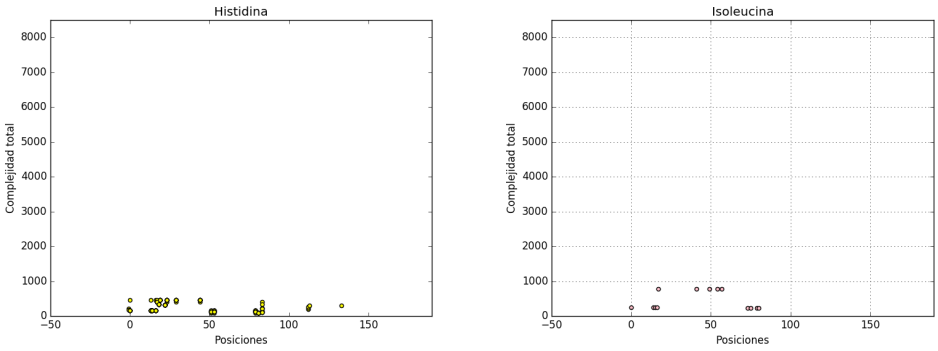


(a) Alanina (b) Cisteina

(c) Ácido Aspártico (d) Ácido Glutámico

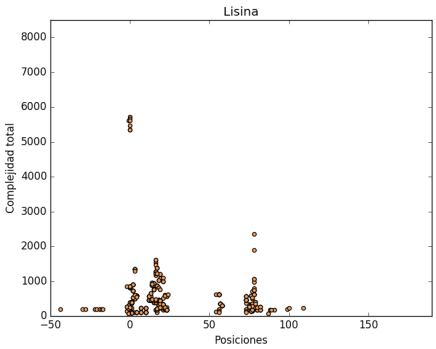
(e) Fenilanina (f) Glicina

Figura 3.9: Se presentan los valores de complejidad total para todas las palabras de contorno de los clusters registrados para los grupos codónicos.

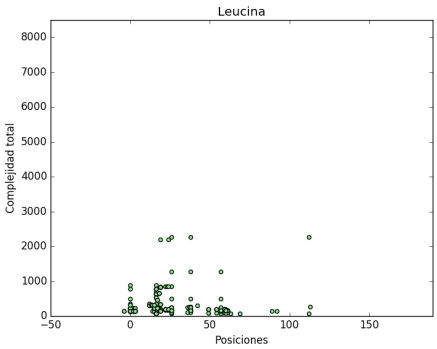


(a) Histidina

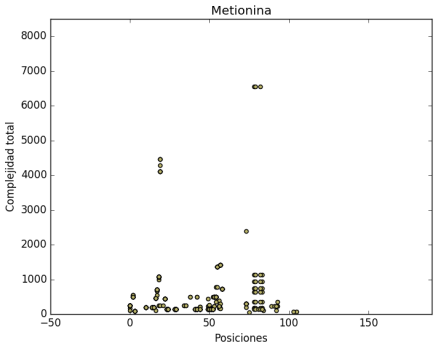
(b) Isoleucina



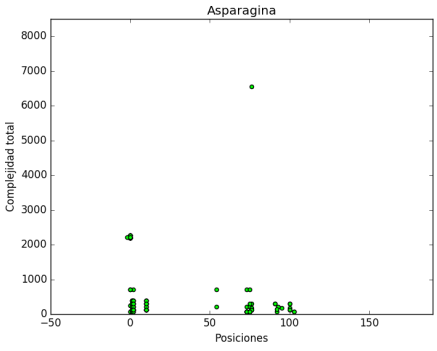
(c) Lisina



(d) Leucina

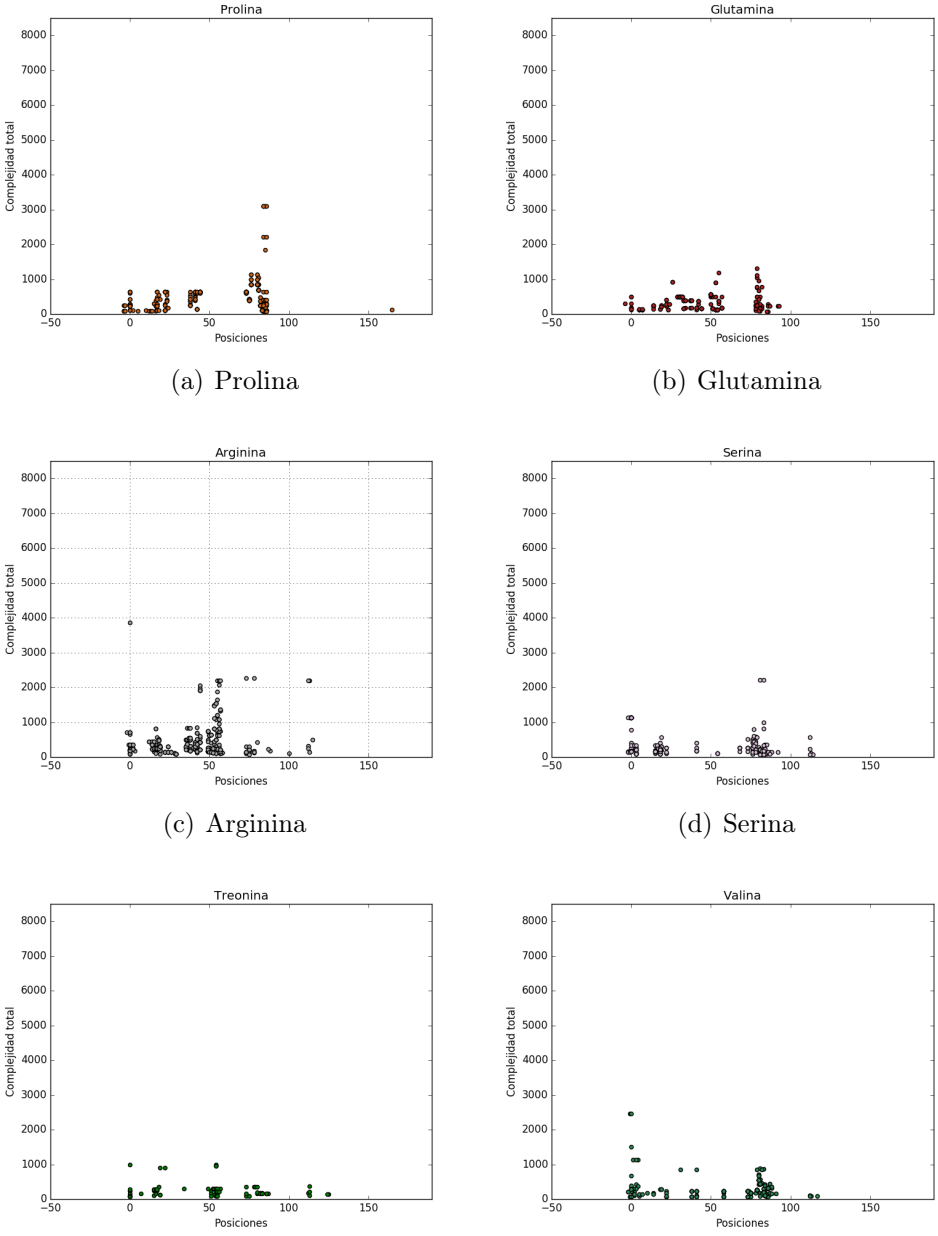


(e) Metionina



(f) Asparagina

Figura 3.10: Se presentan los los valores de complejidad total para todas las palabras de contorno de los clusters registrados para los grupos codónicos.



(a) Prolina

(b) Glutamina

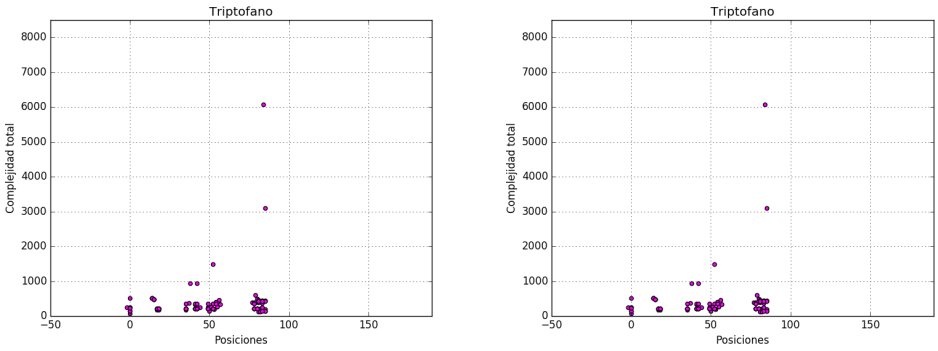
(c) Arginina

(d) Serina

(e) Treonina

(f) Valina

Figura 3.11: Se presentan los los valores de complejidad total para todas las palabras de contorno de los clusters registrados para los grupos codónicos.



(a) Triptofano (b) Tirosina

Figura 3.12: Se presentan los los valores de complejidad total para todas las palabras de contorno de los clusters registrados para los grupos codónicos.

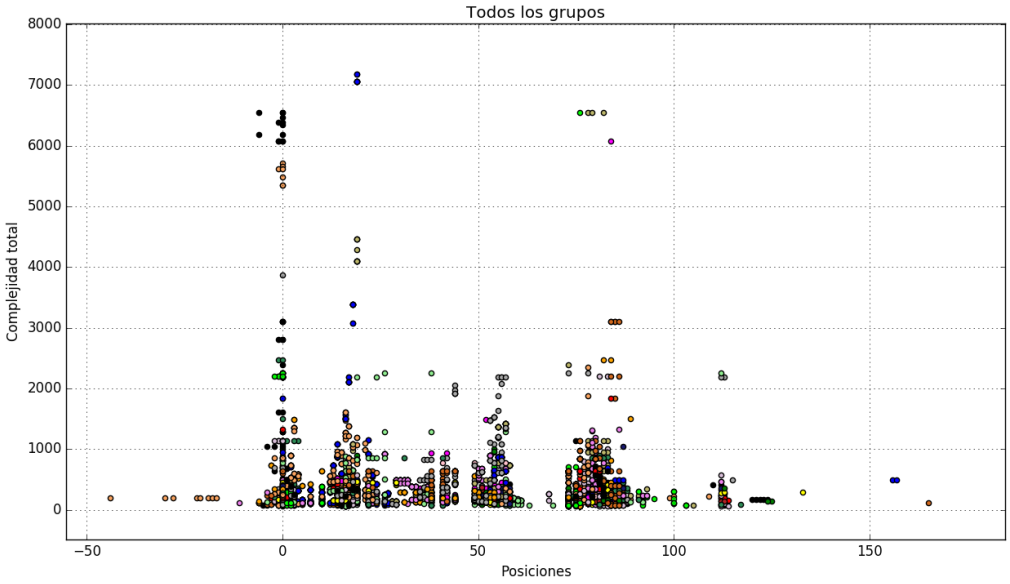


Figura 3.13: Se presentan los los valores de complejidad total para todas las palabras de contorno de los clusters registrados para los grupos codónicos de ARNts

Luego de aplicar los seudocódigos presentados en la sección 3.1.5, sobre cada uno

de los 20 grupos principales de tRNAs que se enlistan en la Tabla A, observamos que los grupos de Alanina, Ácido Glutámico, Arginina tienen alta expresión, es decir que tienen más clusters que los demás grupos.

En cuanto a los valores de complejidad, se evidencia que más de la mitad de los clusters encontrados, son descritos por contornos, la cadena de contorno tiene complejidad de subcadenas que no supera los 2000 factores. Aunque para algunos grupos tales como Alanina, Ácido Glutámico, Metionina, Asparagina y Triptófano existen cluster, en los cuales las cadenas que los describen contienen más de 6000 factores. Esto implica que en las posiciones donde se dan estos altos índices son los que cuentan con la mayor cantidad de secuencias alineadas, y además con formas más irregulares.

También observamos que entre las posiciones 75 y 100, se acumula gran cantidad de clusters de complejidades que no superan los 1000 factores.

3.2.3. Prueba a la metodología

A lo largo de este capítulo se ha presentado la metodología aplicada para la obtención clusters y posterior definición de las cadenas de contorno. En razón a que estas cadenas de contorno, nos permiten tener una representación de la forma de cada uno de los clusters obtenidos, requerimos de la complejidad de total para medir la cantidad de subcadenas de distintas longitudes presentes en tales cadenas.

Dado que distintas cadenas de contorno pueden tener la misma complejidad total, es necesario refinar esta comparación haciendo uso de las proposiciones presentadas en el Capítulo 2, ver sección 2.6.

Basados en las proposiciones de la sección 2.6, dada w, f_w toma valores maximales en un intervalo de longitudes de subcadenas, el cual es definido por propiedades de sus prefijos y sufijos, de la sección 2.5. Definimos para la función f_w, el conjunto $\mathcal{K}_w$ formado por los $1 \leq i \leq |w|$ tales que $f_w(i)$ toma valores maximales.

Rutas en retículo rectangular

Ahora bien, para verificar si la complejidad total permite clasificar las cadenas de contorno, generamos todas las rutas posibles que conectan el origen $(0, 0)$ y $(20, 10)$ utilizando solamente movimientos a derecha y hacia arriba. La Figura 3.14,

muestra el retículo y dos posibles rutas. Estos movimientos los representaremos con los caracteres *1* y *2*, de manera cada ruta queda representada de manera unívoca por una cadena construida con los mismos caracteres que definimos la cadena de contorno de un cluster, ver 3.1.2.

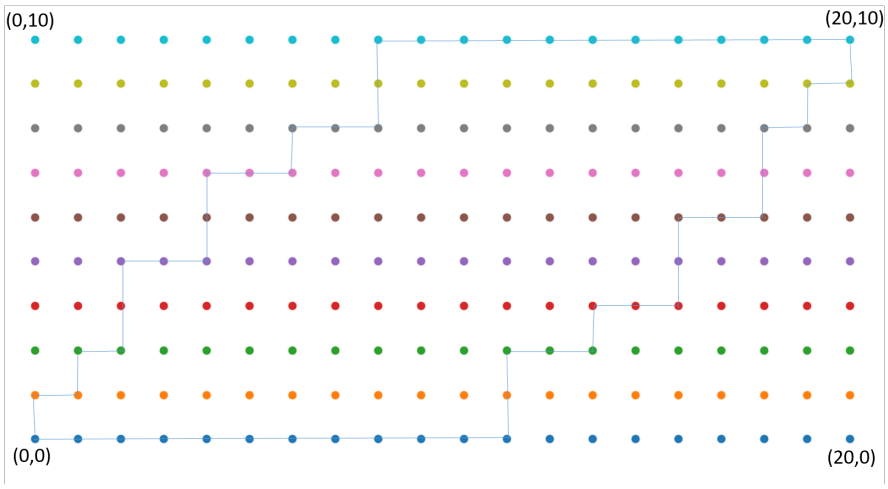


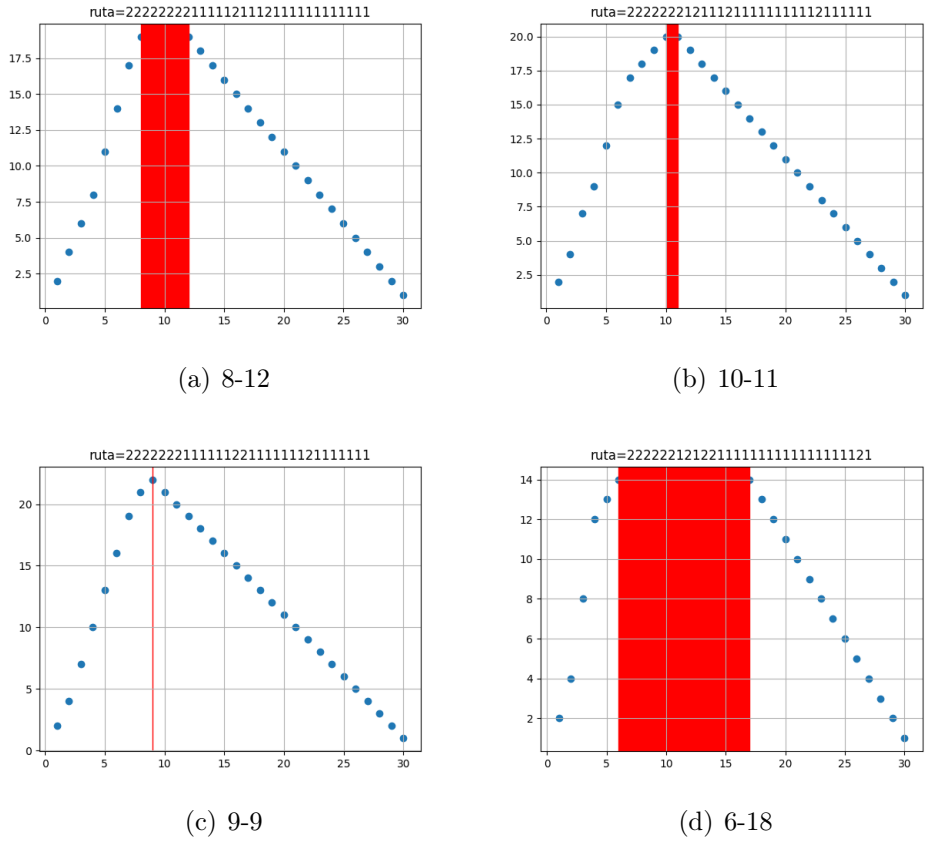
Figura 3.14: En este retículo rectangular las rutas conectan los puntos $(0, 0)$ y $(20, 10)$ mediante pasos unitarios a derecha y hacia arriba

La cantidad de rutas posibles en el retículo corresponde al coeficiente binomial, donde n_1 es el número de pasos a derecha y n_2 el número de pasos hacia arriba.

$$\binom{n_1 + n_2}{n_2} = \binom{30}{10} = 30.045.015.$$

Notemos $\mathcal{P}_r$ a la cadena de caracteres *1* y *2* de la ruta r, calculamos $c(\mathcal{P}_r)$ es decir la complejidad total para cada $\mathcal{P}_r$ y luego buscamos los $\mathcal{K}_{\mathcal{P}_r}$ encontrados para cada valor de complejidad. Lo anterior es con el propósito de usarlos como un criterio adicional de clasificación y comparación.

A continuación se presentan algunas gráficas de la función f_w, en donde resaltamos con una franja roja las longitudes que forman a $\mathcal{K}_{\mathcal{P}_r}$, es decir donde $f_{\mathcal{P}_r}$ alcanza los valores maximales. En el eje x se representan las longitudes que son evaluada en f_w



(a) 8-12

(b) 10-11

(c) 9-9

(d) 6-18

Figura 3.15: Regiones maximales de la función de complejidad.

Con el fin de conocer mejor el comportamiento de $c(\mathcal{P}_r)$ y $\mathcal{K}_{\mathcal{P}_r}$ de las rutas propuestas, en la figura 3.16 muestra como varia la cantidad de posibles valores complejidad (azul) y de intervalos distintos en la función $f_{\mathcal{P}_r}$ (verde), a medida que se avanza en la rutas del retículo.

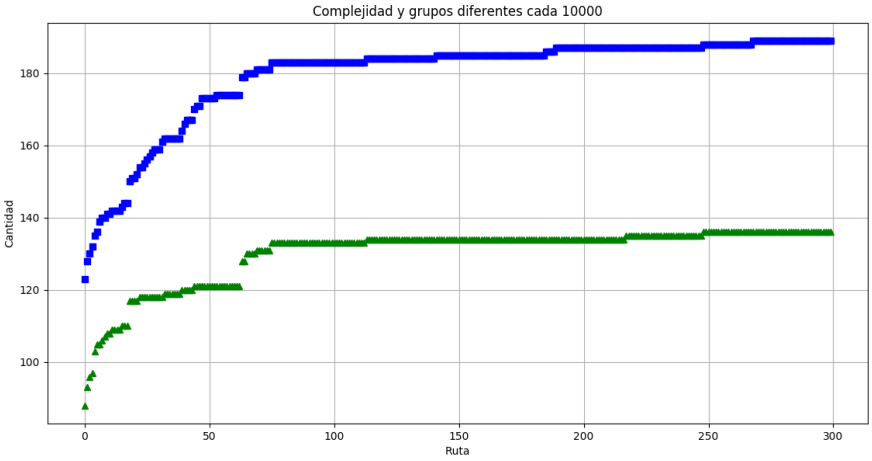


Figura 3.16: En azul la cantidad de valores diferentes de complejidad y en verde la cantidad de grupos diferentes cada 100.000 rutas.

La Figura 3.16 nos muestra como cada 100.000 rutas van apareciendo nuevos valores de $c(\mathcal{P}_r)$ y $\mathcal{K}_{\mathcal{P}_r}$. Como se puede ver estas cantidades presentan cada vez una variación menor que tiende a ser nula. Esto lo utilizamos para conocer como un valor de complejidad y la clasificación mediante el uso de los conjuntos $\mathcal{K}_{\mathcal{P}_r}$, pueden servir como un elemento diferenciador y ser usados como clasificadores de cadenas. En el Apéndice A, se presenta la tabla de los diferentes valores de complejidad encontrados para las rutas generadas, cuantas de ellas tienen el mismo valor de complejidad total y cuantos grupos diferentes se presentan en ellos.

Tasa de falsos positivos

Para cada valor de complejidad total k, buscamos sus correspondientes rutas $\mathcal{P}_r$, tales que $c(\mathcal{K}_{\mathcal{P}_r}) = k$ y los distintos conjuntos $\mathcal{K}_{\mathcal{P}_r}$.

Procediendo de esta forma buscamos las cadenas que no corresponden a contornos de cluster, seleccionadas de rutas del retículo, y les aplicamos el criterio de clasificación usando los conjuntos de clasificación $\mathcal{P}_w$, donde w fueron cadenas de cluster originales. Encontramos la *Tasa de Falsos Positivos* (TFP), es decir los elementos no validos que fueron clasificados como clusters equivalente a

$$TFP = \frac{\text{No. de rutas cluster en el retículo}}{\text{No. de rutas en el retículo}} = \frac{1148}{30.045.015} = 3{,}82\times10^{-5} \sim 4\times10^{-5}.$$

Este resultado indica que bajo las variables utilizadas mediante los valores de la complejidad total y las longitudes sus subcadenas para las que f_w toma valores maximales, en un escenario particular simulado, que el método eficientemente permite encontrar una tasa de falsos positivos baja. Sin embargo escenarios aleatorios adicionales, se sugiere, deben ser utilizados para evaluar diferentes longitudes de cadenas.

Conclusiones

Conclusiones de la metodología

- Al relacionar las cadenas que son alineadas a los ARNts, con las posiciones de inicio de los alineamientos nuestro método permite fragmentar las acumulaciones de alineamientos en clusters de menor tamaño a los analizados por Blockbuster, lo cual se ve reflejado en eficiencia computacional. Gracias a este procedimiento la representación de los contornos de estos clusters se puede realizar con cadenas binarias, dado que solamente se presentan movimientos a derecha y arriba.
- La representación de cada cluster mediante una cadena de contorno, permite que la comparación de la expresión de posiciones de los ARNts se realice solamente entre cadenas binarias, que en pocas palabras contienen la información de cuantas posiciones son mapeadas al ARNt de referencia, cuantas están alineadas a esas posiciones, cuales son las posiciones donde las copias presentan diferencias en su longitud al alternarse los caracteres que describen la ruta del contorno.
- La complejidad de subcadenas es un índice que permite analizar la riqueza de factores que se presentan en las cadenas de contorno, de manera que a mayor cantidad de diferencias de longitud entre copias alineadas más complejidad presentará el contorno como se observó en algunos ARNts .

Conclusiones de los resultados

- Se evidencia en las gráficas que las posiciones de mayor expresión, es decir donde se presentan cluster de mayor cantidad de secuencias alineadas, corresponden con los valores de complejidad de subcadenas.
- Los resultados obtenidos mediante la caracterización de los clusters presentada a lo largo de esta tesis, son coincidentes con los resultados obtenidos

con Blockbuster.

- Se observa que las posiciones donde se concentra la mayor cantidad de clusters para los 20 grupos codones se encuentran en las posiciones entre las posiciones 75 y 100, aunque con complejidad de subpalabras baja en comparación con la que se encuentra en la posición 0, donde se encuentra clusters de valores de complejidad que superan los 6000 factores.
- El incorporar como elemento de clasificación adicional a la complejidad, el conjunto de valores que maximizan a la complejidad por subcadenas, arroja una tasa baja de falsos positivos en un experimento generado de manera aleatoria y longitud de cadenas fija. Por lo tanto, dado que se puede perder generalidad con el resultado obtenido, se debe evaluar la metodología con otros escenarios.

Apéndice A

Valores de la complejidad total y conjuntos de clasificación

La tabla que se presenta a continuación corresponde a la cantidad de rutas que recorren el retículo que se forma entre el origen $(0, 0)$ y $(20, 10)$ y su respectiva cantidad de conjuntos $\mathcal{K}_{\mathcal{P}_r}$, ver sección 3.14.

$c(\mathcal{P}_r)$	Número de rutas	Número de conjuntos ($\mathcal{K}_{\mathcal{P}_r}$)
86	3	1
134	4	1
135	2	1
155	4	1
156	2	1
157	6	1
158	10	1
159	2	1
174	1	1
177	4	1
178	6	1
179	2	1
180	14	1
181	2	1
191	4	1
193	2	1

$c(\mathcal{P}_r)$	Número de rutas	Número de conjuntos ($\mathcal{K}_{\mathcal{P}_r}$)
195	6	1
196	6	1
197	26	1
198	6	2
199	14	1
200	16	1
201	26	1
202	6	1
206	7	1
210	2	1
211	10	1
212	14	1
213	8	1
214	21	1
215	34	2
216	40	2
218	40	1
219	76	3
220	41	1
221	8	1
225	16	1
226	16	1
228	2	1
229	24	2
230	19	2
231	70	2
232	30	2
233	18	2
234	59	1
235	96	2
236	54	1
237	186	2
238	72	4
239	46	3
240	4	1
241	10	1
242	60	2
243	10	3
244	44	1

$c(\mathcal{P}_r)$	Número de rutas	Número de conjuntos ($\mathcal{K}_{\mathcal{P}_r}$)
245	92	1
246	86	4
247	66	4
248	179	3
249	110	2
250	162	1
251	104	4
252	295	3
253	210	3
254	315	4
255	208	6
256	95	5
257	114	2
258	48	3
259	72	2
260	173	5
261	230	6
262	117	5
263	274	4
264	319	5
265	314	5
266	357	6
267	650	6
268	498	6
269	748	8
270	608	7
271	456	7
272	234	4
273	174	7
274	479	11
275	436	10
276	801	12
277	398	11
278	670	9
279	1069	13
280	1237	8
281	1010	9
282	1418	11

$c(\mathcal{P}_r)$	Número de rutas	Número de conjuntos ($\mathcal{K}_{\mathcal{P}_r}$)
283	1724	13
284	1492	11
285	1589	12
286	1031	12
287	1084	13
288	1165	11
289	1378	14
290	2125	13
291	2842	14
292	1606	12
293	3104	14
294	3213	17
295	4110	14
296	4259	14
297	5167	15
298	3602	16
299	3912	18
300	4608	16
301	3224	17
302	5381	16
303	5928	15
304	6744	15
305	6896	17
306	9678	15
307	10447	17
308	11933	18
309	12006	18
310	10877	17
311	11608	18
312	12724	16
313	13216	18
314	14406	18
315	22455	18
316	21196	18
317	23656	16
318	27268	17
319	31071	19
320	29911	21

$c(\mathcal{P}_r)$	Número de rutas	Número de conjuntos ($\mathcal{K}_{\mathcal{P}_r}$)
321	30179	19
322	33012	17
323	41603	18
324	48045	17
325	48300	17
326	56289	19
327	67302	19
328	67140	18
329	74650	17
330	86403	18
331	94820	19
332	103754	19
333	120337	19
334	121193	19
335	145999	19
336	162030	18
337	179344	16
338	194382	16
339	228617	17
340	249564	18
341	256628	18
342	308343	16
343	351196	16
344	378466	17
345	405151	16
346	445204	16
347	523665	15
348	582746	17
349	632221	17
350	684492	17
351	777497	14
352	873406	15
353	914818	14
354	1020229	13
355	1118272	13
356	1195804	13
357	1274865	13
358	1368746	12
359	1417081	12

$c(\mathcal{P}_r)$	Número de rutas	Número de conjuntos ($\mathcal{K}_{\mathcal{P}_r}$)
360	1477990	12
361	1475244	9
362	1442437	10
363	1398236	10
364	1323215	10
365	1174436	10
366	1073910	9
367	937551	8
368	782227	8
369	631435	8
370	485472	7
371	346651	6
372	234169	5
373	133346	5
374	60068	5
375	23384	4
376	7382	2
377	1104	3
378	192	1

Bibliografía

[1] A. V Aho y M. J. Corasick. "Efficient string matching: an aid to bibliographic search". En: *Communications of the ACM* 18.6 (1975), págs. 333-340.

[2] A. Barski y col. "High-resolution profiling of histone methylations in the human genome." En: *Cell* 129.4 (2007), págs. 823-37.

[3] D. Beauquier y M. Nivat. "On translating one polyomino to tile the plane". En: *Discrete & Computational Geometry* 6.1 (1991), págs. 575-592.

[4] C.I. Bermudez. "tRNomics,: Genomic Organization and Processing patterns of tRNAs". Tesis doct. University, Germany, 2010.

[5] I. Chernyakov y col. "Degradation of several hypomodified mature tRNA spceies in Saccharomyes cerevisiae is mediated by Me exonuclease Rat1 and Xrn1". En: *GENES DEVELOP.* 22 (2008), págs. 1369-1380.

[6] N. Cloonan y col. "Stem cell transcriptome profiling via massive-scale mRNA sequencing." En: *Nat Methods* 5.7 (2008), págs. 613-9.

[7] C Cole y col. "Filtering of deep sequencing data reveals the existence of abundant Dicer-dependent small RNAs derived from tRNAs". En: *RNA* 15 (2009), págs. 2147-2160.

[8] A. Colosimo y A. De Luca. "Special factors in biological strings". En: *Journal of theoretical biology* 204.1 (2000), págs. 29-46.

[9] L. A. Copela y col. "The La protein functions redundantly with tRNA modification enzymes to ensure tRNA structural stability". En: *RNA* 12 (abr. de 2006), págs. 644-654.

[10] S. Findeiss y col. "Traces of post-transcriptional RNA modifications in deep sequencing data". En: *Biol. Chem.* 392 (abr. de 2011), págs. 305-313.

[11] M. R. Garcia-Silva y col. "A population of tRNA-derived small RNAs is actively produced in Trypanosoma cruzi and recruited to specific cytoplasmic granules". En: *Mol Biochem Parasitol* (feb. de 2010).

[12] H. J. Haiser y col. "Developmentally regulated cleavage of tRNAs in the bacterium Streptomyces coelicolor". En: *Nucleic Acids Res.* 36 (feb. de 2008), págs. 732-741.

[13] D. Haussecker y col. "Human tRNA-derived small RNAs in the global regulation of RNA silencing". En: *RNA* 1 (2010), págs. 1-5.

[14] S. Hoffmann y col. "Fast Mapping of Short Sequences with Mismatches, Insertions and Deletions Using Index Structures". En: *PLoS Comp. Biol.* 5.9 (2009), e1000502.

[15] L-C. Hsieh y col. "Uncovering small RNA-mediated responses to phosphate deficiency in Arabidopsis by deep sequencing." En: *Plant Physiol* 151.4 (2009), págs. 2120-32.

[16] Wolfgang Huber, Joern Toedling y Lars M. Steinmetz. "Transcript mapping with high-density oligonucleotide tiling arrays". En: *Bioinformatics* 22 (2006), págs. 1963-1970.

[17] C. Jöchl y col. "Small ncRNA transcriptome analysis from *Aspergillus fumigatus* suggests a novel mechanism for regulation of protein-synthesis". En: *Nucleic Acids Res.* 36 (2008), págs. 2677-2689.

[18] D. S. Johnson y col. "Genome-wide mapping of in vivo protein-DNA interactions." En: *Science* 316.5830 (2007), págs. 1497-502.

[19] N.C. Jones y P. Pevzner. *An Introduction to Bioinformatics Algorithms.* A Bradford book. London, 2004.

[20] D. Langenberger y col. "Evidence for human microRNA-offset RNAs in small RNA sequencing data". En: *Bioinformatics* 25.18 (sep. de 2009), págs. 2298-2301.

[21] D. Langenberger y col. "Evidence for human microRNA-offset RNAs in small RNA sequencing data". En: *Bioinformatics* 25.18 (2009), pág. 2298.

[22] Y S Lee y col. "A novel class of small RNAs: tRNA-derived RNA fragments (tRFs)". En: *Genes Dev.* 23 (2009), págs. 2639-2649.

[23] Y. Li y col. "Stress-induced tRNA-derived RNAs: a novel class of small RNAs in the primitive eukaryote *Giardia lamblia*". En: *Nucleic Acids Res* 36 (2008), págs. 6048-6055.

[24] R. Lister y col. "Highly integrated single-base resolution maps of the epigenome in Arabidopsis." En: *Cell* 133.3 (2008), págs. 523-36.

[25] M. Lothaire. *Applied Combinatorics on Words (Encyclopedia of Mathematics and its Applications).* 1.ª ed. Cambridge University Press, 2005.

[26] M. Margulies y col. "Genome sequencing in microfabricated high-density picolitre reactors." En: *Nature* 437.7057 (2005), págs. 376-80.

[27] J. Marioni y col. “RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays.” En: *Genome Res* 18.9 (2008), págs. 1509-17.

[28] A B. Massé y col. “Two infinite families of polyominoes that tile the plane by translation in two distinct ways”. En: *Theoretical Computer Science* 412.36 (2011), págs. 4778-4786.

[29] A. Mortazavi y col. “Mapping and quantifying mammalian transcriptomes by RNA-Seq.” En: *Nat Methods* 5.7 (2008), págs. 621-8.

[30] U. Nagalakshmi y col. “The transcriptional landscape of the yeast genome defined by RNA sequencing.” En: *Science* 320.5881 (2008), págs. 1344-9.

[31] K. Nakanishi y O. Nureki. “Recent progress of structural biology of tRNA processing and modification”. En: *Mol. Cells* 19 (abr. de 2005), págs. 157-166.

[32] J. L. Ramírez, G. N. Rubiano y R. de Castro. “A Generalization of the Fibonacci Word Fractal and the Fibonacci Snowflake”. En: *CoRR* abs/1212.1368 (2014).

[33] F. Sanger, S. Nicklen y A.R. Coulson. “DNA sequencing with chain-terminating inhibitors”. En: *PNAS* 74 (1977), págs. 5463-5467.

[34] F. Sanger y col. “Nucleotide sequence of bacteriophage phi-X174 DNA”. En: *Nature* 265 (1977), págs. 687-695.

[35] S.C. Schuster. “Next-generation sequencing transforms todayś biology”. En: *Nature methods* 5.1 (2008), págs. 16-18.

[36] J. Shendure y col. “Accurate multiplex polony sequencing of an evolved bacterial genome.” En: *Science* 309.5741 (2005), págs. 1728-32.

[37] R. Strausberg y S. Levy. “Promoting transcriptome diversity.” En: *Genome Res* 17.7 (2007), págs. 965-8.

[38] D. M. Thompson y R. Parker. “The RNase Rny1p cleaves tRNAs and promotes cell death during oxidative stress in Saccharomyces cerevisiae”. En: *J. Cell Biol.* 185 (abr. de 2009), págs. 43-50.

[39] D. M. Thompson y col. “tRNA cleavage is a conserved response to oxidative stress in eukaryotes”. En: *RNA* 14 (oct. de 2008), págs. 2095-2103.

[40] G. Vivó-Truyols y col. “Automatic program for peak detection and deconvolution of multi-overlapped chromatographic signals: Part II: Peak model and deconvolution algorithms”. En: *J. Chromatography A* 1096 (2005), págs. 146-155.

[41] Z. Wang, M. Gerstein y M. Snyder. “RNA-Seq: a revolutionary tool for transcriptomics.” En: *Nat Rev Genet* 10.1 (2009), págs. 57-63.

[42] S. Yamasaki y col. “Angiogenin cleaves tRNA and promotes stress-induced translational repression.” En: *J Cell Biol* 185.1 (2009), págs. 35-42.

[43] Q. Zhao y col. “Transcriptome-guided characterization of genomic rearrangements in a breast cancer cell line.” En: *Proc Natl Acad Sci U S A* 106.6 (2009), págs. 1886-91.