INSTITUTO POTOSINO DE INVESTIGACIÓN CIENTÍFICA Y TECNOLÓGICA, A.C.

POSGRADO EN CONTROL Y SISTEMAS DINÁMICOS

Análisis matemático para la caracterización de ARNs largos no codificantes

Tesis que presenta

David Iván Hernández Granados

Para obtener el grado de

Doctor en Control y Sistemas Dinámicos

Directores de la Tesis:

Dr. Hugo Cabrera Ibarra
Dra. Lina Raquel Riego Ruiz

San Luis Potosí, S.L.P., 05 de diciembre de 2025

Constancia de aprobación de la tesis

La tesis **Análisis matemático para la caracterización de ARNs largos no codificantes** presentada para obtener el Grado de Doctor en Control y Sistemas Dinámicos fue elaborada por **David Iván Hernández Granados** y aprobada el **05 de diciembre de 2025** por los suscritos, designados por el Colegio de Profesores de la División de Control y Sistemas Dinámicos del Instituto Potosino de Investigación Científica y Tecnológica, A.C.

Dr. Daniel Alejandro Melchor Aguilar
Presidente

Dr. Hugo Cabrera Ibarra
Secretario

Dra. Lina Raquel Riego Ruiz
Sinodal

Dr. Javier Flavio Vigueras Gómez
Sinodal

Dr. David Antonio Lizárraga Navarro
Sinodal

Créditos Institucionales

Esta tesis fue elaborada en la División de Control y Sistemas Dinámicos del Instituto Potosino de Investigación Científica y Tecnológica, A.C., bajo la dirección del Dr. Hugo Cabrera Ibarra (DCS) y de la Dra. Lina Raquel Riego Ruiz (DBM).

Durante la realización de este trabajo el autor recibió una beca académica de la Secretaría de Ciencia, Humanidades, Tecnologías e Innovación asignada con el número 619730 y del Instituto Potosino de Investigación Científica y Tecnológica, A. C.

(Página en Blanco que se va a utilizar para colocar la copia del acta de examen.)

Con mucho cariño y especial dedicación para mi hija Ruth.
Que esta tesis te quede como una prueba fehaciente de que con esfuerzo y dedicación una persona puede sobreponerse a sus malas decisiones y a las adversidades.

Agradecimientos

• A mis asesores el *Dr. Hugo Cabrera Ibarra* y la *Dra. Lina Raquel Riego Ruiz*, por brindarme su apoyo total, su buen ejemplo y por invertir en mí no sólo su conocimiento sino también algo de su tiempo. Especialmente: al Dr. Hugo agradecer su voto de confianza en mí y su apoyo constante, con el cual he podido abrirme camino en el ámbito de la ciencia y de la divulgación. A la Dra. Lina agradecer sus palabras de aliento y motivación que me invitan a seguir adelante.

• A *mis dos familias*, fuente de apoyo constante e incondicional. Especialmente, agradecer *a mis padres*, ya que sin su ejemplo, esfuerzo, comprensión y amor, no hubiera tenido la oportunidad de ser una persona de bien y mucho menos tener una formación académica.

• A *mi esposa*, por su apoyo incondicional, paciencia y amor. Especialmente, agradecer sus palabras de aliento, sus retos, su buen ejemplo y su dedicación, lo cual me motiva siempre a seguir adelante y a esforzarme un poco más día a día con el fin de estar a su nivel.

• Por último y no menos importante. Quiero agradecer a los doctores: *Dr. Daniel Alejandro Melchor Aguilar*, *Dr. David Antonio Lizárraga Navarro* y *Dr. Javier Flavio Vigueras Gómez*, por sus valiosos comentarios, conocimientos, disponibilidad y sobre todo por la enriquecedora discusión sobre los por menores y alcances de mi trabajo de tesis, permitiéndome con ello madurar mi propuesta de investigación y visualizar otras perspectivas.

Contenido

Lista de figuras

Lista de tablas

Resumen

Los ARNs no codificantes (ARNs-nc) están involucrados en diversos procesos biológicos, por lo que su identificación y caracterización funcional es una prioridad. Entre ellos, se encuentran los ARNs largos no codificantes (ARNs-lnc). Recientemente se ha demostrado que estos regulan diversos procesos celulares, como el desarrollo celular, la respuesta al estrés y la regulación transcripcional. El identificar ARNs-lnc y cuál es su función a nivel celular es una ardua tarea que se lleva a cabo en los laboratorios, para la cual se necesita de tiempo y recursos. Así, la continua identificación de nuevos ARNs-lnc remarca la necesidad de métodos fiables para su detección, siendo el análisis estructural una herramienta fundamental. Buscando potenciar la búsqueda e identificación de ARNs-lnc, recientemente se han empleado programas y herramientas computacionales; un ejemplo de ello es LncFinder, el cual incluye una gran base de ARNs-lnc de humanos, ratones, pollos, entre otros organismos. Sin embargo, muchas de estas herramientas se enfocan en analizar solo ARNs de forma preferencial; por ejemplo, en mamíferos se prioriza el análisis de los ARNs de humanos.

Por esta razón, en esta tesis se expone un método que tiene como base elementos matemáticos y computacionales que busca caracterizar grupos de ARNs, y cuyos resultados pueden posteriormente validarse de forma experimental. Para ello, se propuso un algoritmo basado en gráficas de árbol con raíz para representar y comparar estructuras secundarias del ARN, con la finalidad de establecer una caracterización partiendo de tres conjuntos de ARNs, de los cuales nos interesa una característica de su estructura relacionada con su función. Primero, el algoritmo analiza los primeros dos conjuntos para extraer información estructural; es decir, se buscan similitudes estructurales entre ambos conjuntos. Después, el algoritmo analiza si dichas similitudes están presentes, o no, en el tercer conjunto. Finalmente, se establece qué elementos del tercer conjunto tienen la característica funcional de interés. Además, dicho algoritmo fue aplicado para analizar ARNs-lnc pertenecientes a dos organismos: *Saccharomyces cerevisiae* y *Homo sapiens*. Por medio de este método, se lograron identificar correctamente secuencias de ARNs-lnc en el 90 % de los casos. Estos resultados muestran que el análisis estructural basado en gráficas ofrece una metodología complementaria para la identificación de ARNs-lnc y puede complementar las herramientas existentes que se basan en analizar los nucleótidos que componen la secuencia como lo son lncFinder o PreLnc. Así mismo, cabe mencionar que este algoritmo puede ser utilizado para detectar similitudes estructurales en otros tipos de organismos y en la búsqueda de otras características que relacionen su estructura con su funcionalidad. Por ejemplo, estudios recientes han mostrado que las células tumorales pueden secretar ARNs-lnc en los fluidos biológicos de los seres humanos, formando ARNs-lnc circulares que pueden servir como biomarcadores en el cáncer. Para este caso, el algoritmo propuesto se podría aplicar para la identificación de nuevos ARNs-lnc que presenten similitudes estructurales con los ARNs-lnc circulares asociados con los tumores malignos.

Palabras clave: ARN; ARNs-lnc; estructura secundaria del ARN; gráficas de árbol con raíz; algoritmo; *Saccharomyces cerevisiae*.

Abstract

Non-coding RNAs (ncRNAs) are involved in many biological processes, making their identification and functional characterization a priority. Among them are long non-coding RNAs (lncRNAs). Recently, it has been shown that they regulate diverse cellular processes, such as cell development, stress response, and transcriptional regulation. Identifying lncRNAs and determining their functions at the cellular level is an arduous task carried out in laboratories, requiring both time and resources. Thus, the continuous identification of new lncRNAs underscores the need for reliable methods for their detection, with structural analysis offering insightful information. Currently, lncRNAs are identified using tools such as LncFinder, whose database has a large collection of lncRNAs from humans, mice, and chickens, among others. However, some of these tools focus on analyzing only RNAs preferentially; for example, in mammals, the analysis of human RNAs is prioritized.

For this reason, this thesis presents a method based on mathematical and computational elements, whose results can subsequently be validated experimentally. For this purpose, an algorithm based on rooted tree graphs was proposed to represent and compare RNA secondary structures, with the aim of establishing characterization based on three sets of RNAs sequences, of which we are interested in a particular functional characteristic. First, the algorithm analyzes the first two sets to extract structural information; that is, it searches for similar structures between both sets. Next, the algorithm analyzes whether these similarities are present in the third set or not. Finally, it is determined which elements of the third set have the functional characteristic of interest. Besides, this algorithm was applied to analyze RNAs belonging to two organisms: *Saccharomyces cerevisiae* and *Homo sapiens*. Using this method, lncRNA sequences were correctly identified in 90 % of cases. These results show that graph-based structural analysis offers a complementary methodology to identify lncRNAs and can complement existing sequence-based tools such as lncFinder or PreLnc. Likewise, it is worth mentioning that this algorithm can be used to detect structural similarities in other types of organisms and to search for other functional characteristics. For example, recent studies have shown that tumor cells can secrete lncRNAs into human biological fluids, forming circulating lncRNAs that can serve as cancer biomarkers. In this case, the proposed algorithm could be applied to identify novel lncRNAs that exhibit structural similarities to circular lncRNAs associated with malignant tumors.

Key Words: RNA; lncRNAs; secondary structure; rooted tree graphs; algorithm; *Saccharomyces cerevisiae*.

Capítulo 1

Introducción

En los últimos años, la biología ha incorporado algunas herramientas matemáticas con el fin de resolver algunos problemas y con ello, ha motivado desarrollos teóricos en muchos campos de las matemáticas y la computación. En particular, en la teoría de gráficas, el análisis y la topología. Dado que la biología actualmente ha tenido muchos avances, las oportunidades para emplear diversas áreas de las matemáticas al estudio de los sistemas biológicos se han incrementado notablemente. Especialmente, la estadística y la computación han sido aspectos claves en el desarrollo de la biología moderna, por ejemplo: en la modelización biomolecular o en la genómica.

Dentro de la biología, el estudio del ácido ribonucleico (ARN) ha sido importante debido a su papel clave dentro de las etapas intermedias de la síntesis proteica. El ARN puede replicarse a sí mismo, actuar como una enzima y es un componente clave en la síntesis de proteínas. Esta molécula está formada por una sola cadena de ribonucleótidos, que pueden formar cadenas dobles entre sí a través de puentes de hidrógeno entre las bases nitrogenadas. Es de gran interés ilustrar estas redes de interacción, conocidas como estructuras secundarias del ARN, las cuales se pueden modelar como una gráfica [1]. El representar estas redes de interacción mediante gráficas ha sido de gran utilidad en el análisis estructural del ARN, lo que ha permitido muestrear su espacio de plegamiento, predecir sus pliegues tridimensionales y aplicar aspectos combinatorios para diseñar nuevos ARNs *in vitro* [1, 2]. Por ejemplo, Liu y colaboradores en [3] estudian los R-loops, los cuales son una clase de estructura no canónica de ácidos nucleicos que se suelen formar durante la transcripción. Para ello, asocian a una estructura secundaria de ARN una gráfica de árbol con raíz y un polinomio único denominado tree-polynomial. Demostraron que existe una fuerte correlación entre la suma de los coeficientes de dicho polinomio y la probabilidad experimental de la formación de un R-loop. Otro ejemplo es el caso de Gan y colaboradores [4] que desarrollaron una representación gráfica bidimensional para describir y estimar el tamaño del repertorio de las estructuras secundarias del ARN. Usaron para ello gráficas de árbol con raíz, con las cuales describen motivos estructurales de las gráficas y señalan qué topologías, que no están registradas en las bases de datos del ARN, podrían servir como modelo para el diseño de nuevas secuencias de ARN. En las últimas décadas, el interés en la estructura, función y diseño del ARN, ha crecido significativamente; especialmente, con el descubrimiento de ARNs reguladores no codificantes (ARNs-nc) [5, 6], incluidos los micro ARNs (ARNs-mi) y los ARNs largos no

codificantes (ARNs-lnc). En los últimos años, se han encontrado relaciones entre el ARN y muchas enfermedades humanas [6, 7, 8], lo que abre una nueva ventana de oportunidad para la detección de enfermedades y sus respectivas terapias [9]. Actualmente, el análisis gráfico se ha incorporado como una herramienta útil para extraer características globales de las biomoléculas con la finalidad de caracterizarlas [1, 10, 11]. Por ejemplo, para la detección de ARNs-lnc, Siyu *et al.* [12] desarrollaron LncFinder, la cual es una plataforma integrada basada en algoritmos de aprendizaje automático que incluye un predictor de ARNs-lnc con buen desempeño para la detección de ARNs-lnc de organismos como humanos, ratones, pollos, entre otros. Otra de ellas, propuesta por Cao *et al.* [13], es PreLnc, el cual utiliza transcritos de ARNs-lnc y ARNs mensajeros validados para construir sus modelos de predicción mediante selección de características y clasificadores. Para ello, analizan la composición de los trinucleótidos de las transcripciones de diferentes especies y concluyen que su enfoque es prometedor para la detección de ARNs-lnc para la transcriptómica a gran escala.

Por otro lado, se ha establecido que los ARN-nc están involucrados en diversos procesos celulares. Por ejemplo, se sabe que juegan un papel muy importante en la diferenciación molecular, la elección del linaje celular y la organogénesis [14, 15, 16]. Bernstein y colaboradores en [17] sugieren que los ARNs-nc transcripcionales están más estrechamente relacionados con los procesos biológicos de lo que se creía anteriormente. Para su estudio, se han establecido dos categorías [18]: ARNs-nc con menos de 200 nucleótidos, por ejemplo los ARNs pequeños (ARNs-p), y ARNS-nc con más de 200 nucleótidos conocidos como ARNs largos no codificantes (ARNs-lnc). Durante la última década, se ha demostrado que la transcripción generalizada de genomas eucarióticos produce una gran cantidad de ARNs-lnc [19]. Sin embargo, no existen características claras que nos permitan identificarlos ya que entre organismos no conservan la secuencia de nucleótidos. Por lo general, solo se sabe que poseen más de 200 nucleótidos y que no poseen marcos de lectura abiertos funcionales [20]. Por ello, dada la importancia de los ARNs-lnc, es relevante establecer un método para analizar su estructura y, en caso de ser posible, determinar si alguna subestructura puede caracterizarlos.

El estudio de los ARNs-lnc en levaduras ha sido escaso, comparado con los estudios recientes enfocados en humanos y en otros mamíferos [21]. Yamashita y colaboradores mencionan en [22] que estudiar las levaduras es importante debido a su trazabilidad genética, a su velocidad de crecimiento y a la facilidad con la que se pueden cultivar para realizar experimentos. Además, se sabe que muchos de los procesos celulares se conservan tanto en la levadura *Saccharomyces cerevisiae* como en humanos [23, 24, 25]. En conclusión, estudiar la levadura *S. cerevisiae* es un buen punto de partida para el análisis de organismos más complejos.

Así, siguiendo el enfoque de Gan *et al.* [4], se desarrolló un algoritmo que permite analizar la estructura secundaria del ARN. Para ello, el algoritmo analiza las estructuras secundarias asociadas a conjuntos de secuencias de ARN asignando, a cada una de ellas, una gráfica de árbol con raíz, la cual es codificada con una sucesión de puntos y paréntesis. Después, el algoritmo compara los conjuntos determinando las subestructuras que comparten. Finalmente, el resultado de este análisis es utilizado para establecer una conclusión, la cual indica si una secuencia de ARN pertenece a un conjunto con cierta característica.

Objetivos:

Realizar un análisis detallado de las gráficas de árbol con raíz asociadas a algunas cadenas de ARNs-lnc con el fin de extraer información estructural. Para con ello, buscar si existen estructuras que los relacionen. Analizar si estas estructuras están presentes en otros grupos de ARNs, buscando hacer una clasificación de los ARNs-lnc por medio de su estructura, para con ello, generar algunas predicciones para ARNs no clasificados.

Así, en esta tesis se busca establecer un método que permita desde un enfoque teórico, es decir, que prescinda de un laboratorio, determinar si una secuencia de ARN puede tener la característica funcional de ser un ARN-lnc.

Para ello, en el Capítulo 2 se expondrán algunos detalles y conceptos básicos de biología que servirán como base para entender este trabajo, enfocándose en el ARN, los ARNs-lnc y su estructura secundaria. Luego, en el Capítulo 3 se introducirá un enfoque basado en gráficas de árbol con raíz que permitirá estudiar una secuencia de ARN desde un punto de vista matemático, mostrando la utilidad de las cadenas en notación punto-paréntesis simplificado, *NPPS*. Con el enfoque antes mencionado, en el Capítulo 4 se implementará un algoritmo que permitirá identificar similitudes estructurales entre secuencias de ARNs y determinar si una secuencia de ARN es potencialmente un ARN-lnc; del mismo modo, se mostrará un ejemplo práctico de su funcionamiento. Después, en el Capítulo 5 se empleará el algoritmo para tratar de identificar similitudes estructurales de los organismos: *Saccharomyces cerevisiae* y *Homo sapiens*. Además, en el Capítulo 6 se mostrará el algoritmo y su uso. Posteriormente, en el Capítulo 7 se establecerán las conclusiones y trabajo a futuro referentes a esta tesis. Por último, se encontrarán cinco Apéndices: en el Apéndice A se mostrará la productividad derivada de este trabajo de tesis. Mientras que, en el apéndice B se explorará un breve análisis de la estructura primaria del ARN. Luego, en el Apéndice C se mostrará una guía para utilizar la plataforma NUPACK web, con la cual se pueden generar las estructuras secundarias de los ARNs a estudiar. Mientras que, en el Apéndice D se mostrarán las secuencias de los ARNs analizados provenientes de *H. sapiens*. Y finalmente, en el Apéndice E se mostrará el pseudocódigo referente al algoritmo propuesto, con la finalidad de que se pueda implementar en diversos lenguajes de programación.

Capítulo 2

Preliminares

En el presente capítulo se expondrán algunos detalles y conceptos básicos de biología que servirán como base para el desarrollo de esta tesis.

2.1. ADN y ARN

La biología es la ciencia que estudia las características de los seres vivos, así como su origen, su evolución y su adaptación. Su fin es establecer una ley general que defina la estructura y la dinámica funcional común en todos los seres vivos [26]. Para su estudio, la biología se divide en subdisciplinas según la escala y los tipos de organismos que se estudian. Por ejemplo: la biología molecular que se encarga de estudiar la estructura física y química de las biomoléculas [27].

Se considera que la biología molecular nace en 1953 gracias a las investigaciones de Rosalind Franklin, así como de James Watson y Francis Crick, quienes lograron establecer un modelo de estructura secundaria para el ácido desoxirribonucleico (ADN). A partir de la publicación de dichas investigaciones, el estudio de los mecanismos moleculares y el procesamiento de la información genética presente en el ADN se intensificó [26, 28].

Actualmente, se sabe que el ADN contiene las instrucciones genéticas que permiten generar una nueva célula o un nuevo organismo. Cada molécula de ADN, presente en los cromosomas de las células, está constituida por genes, los cuales contienen la información para regular la elaboración de proteínas, misma que debe ser transcrita selectivamente a moléculas de ácido ribonucleico (ARN), para que posteriormente algunas de ellas sean traducidas a proteínas [28]. Si bien el ADN contiene las instrucciones genéticas, el ARN se encarga de que dichas instrucciones sean comprendidas por las células.

Desde un punto de vista químico, el ADN es una molécula bicatenaria formada por dos cadenas simples de desoxirribonucleótidos, las cuales incluyen: un grupo fosfato, un grupo de azúcar y una de las cuatro bases nitrogenadas: adenina (A), citosina (C), guanina (G) y timina (T), ver Figura 2.1. Para que el ADN adquiera su estructura bicatenaria se debe llevar a cabo un emparejamiento químico, por medio de puentes de hidrógeno, en el cual la base A se complementa siempre con la base T (A-T), mientras que la base C se complementa con la base G (C-G) [26].

Debido a la evolución de diversas metodologías moleculares como la secuenciación y la cristalografía, ha sido posible definir que el ARN es una molécula monocatenaria, que sigue una dirección en sentido $5'$ a $3'$, formada por una cadena simple de ribonucleótidos, la cual está formada por: ribosa, un fosfato y una de las cuatro bases nitrogenadas: adenina (A), citosina (C), guanina (G) y uracilo (U) [28]. Dado que el ARN es monocatenario tiene la capacidad de formar una gran variedad de estructuras estables, las cuales proporcionan información biológica significativa, dado que las funciones biológicas del ARN dependen de su estructura [29]. La Figura 2.1 muestra una cadena de ARN (en A) y una de ADN (en B), en la cual se puede notar la diferencia entre ellas.

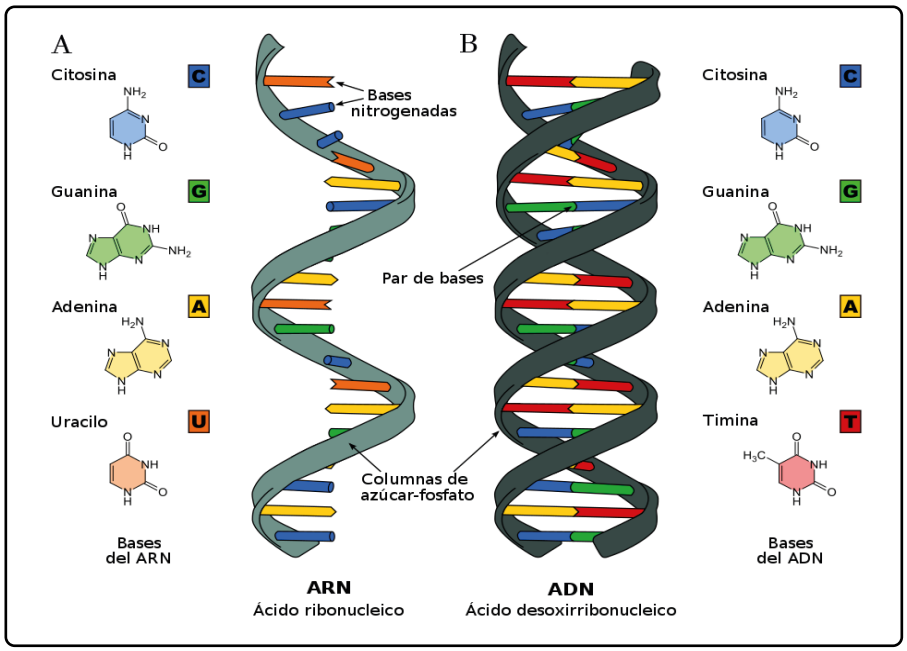


Figura 2.1: A diferencia del ADN (en B), el ARN (en A) es una molécula monocatenaria. Además, difieren en una de sus bases nitrogenadas, en el ADN: timina (T) y en el ARN: uracilo (U). Fuente de la imagen [30].

Existen fundamentalmente tres grupos de ARNs transcritos [29]: el ARN mensajero (ARN-m), el cual contiene la información de los aminoácidos que contendrá una proteína y representa del 3 % al 5 % del ARN, el ARN de transferencia (ARN-t) que representa del 5 % al 7 % del ARN y el ARN ribosomal (ARN-r) que representa del 85 % al 90 % del ARN, de los cuales el ARN-t y el ARN-r forman parte de la maquinaria de traducción del ARN-m a las proteínas [28]. Además, existen otros ARNs que no cumplen con la función de ser mensajeros en la síntesis de proteínas, a los cuales se les denominan ARNs no codificantes (ARNs-nc) [31]. Esto no establece que dichos ARNs no sean importantes; al contrario, actualmente se sabe que los ARNs-nc están involucrados en diversos procesos celulares a nivel epigenético, transcripcional o postranscripcional [32]. Bernstein y colaboradores [17] sugieren que el ARN no codificante está más relacionado con los procesos biológicos de lo que se creía anteriormente. Para su estudio, los ARNs han sido clasificados en dos categorías [18]: 1. ARNs con menos de 200 nucleótidos, como lo son, por ejemplo, los micro ARNs (ARNs-mi) o los ARNs pequeños (ARNs-p), 2. ARNs con más de 200 nucleótidos y que no codifican para proteínas, conocidos como ARNs largos no codificantes (ARNs-lnc).

2.1.1. ARNs largos no codificantes

Los ARNs largos no codificantes (ARNs-lnc) son ARNs que suelen tener más de 200 nucleótidos de longitud. Algunas veces, pueden presentar características similares a las del ARN-m; por ejemplo, tener exones o poseer la cola de poli(A). Los ARNs-lnc se procesan después de la transcripción, pero no codifican para proteínas; o bien, carecen de un marco de lectura abierto [33]. Recientemente, se ha establecido que son importantes en la regulación de procesos muy diversos de la célula; por ejemplo, en la diferenciación celular, en la respuesta al estrés, en la elección del linaje celular, en la regulación transcripcional, en la organogénesis, entre otros [14, 15, 16]. Con ello, la importancia de estudiar los ARNs-lnc se ha incrementado notablemente, dado que las estructuras que los componen y las formas en las que actúan sobre otras moléculas permanecen en la etapa de caracterización [34]. De hecho, la secuenciación y el análisis de genomas humanos y de ratones sugieren que muchos ARNs no codificantes aún quedan por ser descritos y caracterizados [31, 35].

Chowdhary y colaboradores mencionan en [36] algunas técnicas empleadas para tratar de identificar ARNs-lnc. De las cuales, se tienen por un lado las aproximaciones experimentales como lo son, por ejemplo: 1) la tecnología de Microarreglos que permite la detección de ADN o ARN por medio de la adherencia o la hibridación de las moléculas en cuestión en un portaobjetos de vidrio. Los ácidos nucléicos son marcados con algunas etiquetas que se pueden detectar con ayuda de una cámara CCD. 2) Los análisis de RNA-seq, que se basan en secuenciar una parte del genoma enfocándose en el ARN que se transcribe, buscando comparar los patrones de expresión entre las muestras problema y las muestras control. Y 3) el Análisis en Serie de la Expresión Génica (SAGE por sus siglas en inglés), que analiza los ARNs-m que se encuentran presentes en la célula en cierto momento y busca cuantificar la expresión de sus genes, creando con ello diferentes perfiles que determinan qué gen está siendo activado o no ante determinada condición. Por otro lado, se tienen las aproximaciones computacionales como lo son: 1) la búsqueda de los marcos de lectura abiertos (ORFs por sus siglas en inglés), buscando aquellos que tengan una longitud mayor a 100 codones. 2) Los análisis basados en el aprendizaje automático; en los cuales, usando modelos de regresión logística, se buscan características para cuantificar el nivel de codificación de la transcripción, para predecir si será un ARN-lnc o un ARN-m. Algunas características que se toman en cuenta en este tipo de análisis son: la búsqueda de ORFs, el análisis de la preservación de las regiones exónicas o de sus motivos estructurales, la composición de nucleótidos y el uso de codones. Cabe señalar que el análisis de preservación, conocido también como búsqueda de homología, tiene como finalidad establecer una clasificación de los ARNs-lnc basándose en la conservación de las regiones exónicas, su secuencia y su posición genómica [37]. Este tipo de análisis ha servido como base para entender un poco el comportamiento de los ARNs-lnc; por lo cual, es necesario complementar estas aproximaciones buscando robustecerlas y mejorar sus predicciones. Un camino para ello está en una nueva vertiente que relaciona el funcionamiento de un ARN-lnc con su estructura secundaria más que con la secuencia que lo describe [18, 38, 39, 40, 41].

Actualmente, el uso de programas y herramientas computacionales para identificar ARNs-lnc han abierto un área de oportunidad. Sin embargo, la mayoría de estas metodologías aún están en desarrollo y se prioriza analizar solo ARNs de ciertos organismos; por ejemplo,

en mamíferos se prioriza estudiar los ARNs de humanos y de ratones. Un ejemplo de estas herramientas es el programa LncFinder [12], el cual incluye una gran base de ARNs-lnc de humanos, ratones, pollos, entre otros organismos. Sin embargo, muchas de estas herramientas se enfocan en analizar solo ciertos ARNs de forma preferencial; por ejemplo, en mamíferos se prioriza el análisis de los ARNs de humanos. En consecuencia, el estudio de los ARNs-lnc en levaduras ha sido escaso, comparado con los estudios recientes enfocados en humanos y en otros mamíferos [21]. Yamashita y colaboradores mencionan en [22] que estudiar las levaduras es importante debido a su trazabilidad genética, a su velocidad de crecimiento y a la facilidad con la que se pueden cultivar para realizar experimentos. Además, se sabe que muchos de los procesos celulares se conservan tanto en levaduras como en humanos [23, 24, 25]. Aunque existen más de 1000 genomas completos de la levadura *S. cerevisiae*, perteneciente a la familia de hongos *Ascomycetes*, no en todos los organismos se han desarrollado las suficientes herramientas moleculares para poder estudiar, de manera funcional, los distintos procesos. Y menos aún se han desarrollado herramientas matemáticas y de análisis que permitan entender las similitudes y las diferencias que hay en algunos de los procesos biológicos presentes en estos organismos. En conclusión, estudiar la levadura *S.cerevisiae* es un buen punto de partida para el análisis de organismos más complejos.

2.1.2. ARNs y el formato FASTA

Para poder analizar un ARN desde un enfoque teórico, es necesario tener la secuencia que lo describa en formato FASTA, el cual es un formato basado en texto, muy utilizado para representar secuencias de ácidos nucleicos o de péptidos, donde los pares de bases o los aminoácidos se representan usando códigos de una única letra. En el caso del ARN se usan solo las iniciales de las cuatro bases nitrogenadas: A, C, G y U. Además, en dicho formato se incluye el nombre de la secuencia y algunos comentarios, para ello, deben estar en el mismo renglón y ser precedidos por el símbolo mayor que $>$ [42], el Ejemplo 1 muestra la secuencia en formato FASTA del ARN *ZOD1*.

Ejemplo 1. *Secuencia en formato FASTA:*

> Saccharomyces cerevisiae - ZOD1
GCUUCUUGUUACCCCUUUUGGCGCUUUGGAUAAGUAGUUCACUUAGCUAUUUUUUUUU

En esta tesis, para obtener el conjunto de los ARNs para analizar, se extraerán de cada secuencia en formato FASTA, solo el nombre del ARN y la secuencia de nucleótidos que lo describen, refiriéndose a ellos como los datos de la secuencia. El Ejemplo 2 muestra el *ARN170* que se analizará.

Ejemplo 2. *Datos de la secuencia a analizar:*

ARN170: UUCUCUUAUAUACCAUGGCGCUGCAGAUCUAUCCAAAUUCUUCCAUACUGAAAA
UCAUCCAAACGAGAACGAUAGCACCGGUAGUUUUCCUUGCGGCUUAUACGAGGGCCGUG
GUUCGAUUCUGCCCUGGAGCGUAAGGGCAGGAAUGUGCAUUGCAUAAGAUCAUAAU

2.1.3. *Saccharomyces cerevisiae*

La levadura *S. cerevisiae* es un organismo eucariota unicelular, encapsulado por una pared celular compuesta de varias capas que le dan protección al organismo y le ayudan a la detección de las señales ambientales, creando así una interfaz entre la célula y el medio ambiente. Entender su dinámica permite modificar las células de la levadura para hacer de este organismo una mejor herramienta para la biotecnología y la biología sintética [43]. Esta levadura es utilizada ampliamente en el estudio de las células eucariotas y, a menudo, se emplea como una fábrica de células para la producción de proteínas, productos químicos, alimenticios y farmacéuticos, así como en la fabricación de biocombustibles [43, 44]. Algunas de las características importantes de esta levadura son su función como nutriente y su capacidad tanto para producir etanol, como para la fermentación a gran escala [45].

En los últimos años, los ARNs-lnc identificados en la levadura *S. cerevisiae* han aumentado rápidamente, pero, aun así, son muy escasos aquellos en los cuales hay evidencia experimental que valide su función. Entre 2018 y 2019 solo se habían descrito adecuadamente 18 ARNs-lnc para esta especie [21, 46], en 2020 se anexó uno más a la lista [47]. Estos ARNs-lnc están involucrados directamente en los cambios metabólicos, la diferenciación sexual y otros procesos desconocidos [22]. Además, la mayoría de estos ARNs-lnc están implicados en la regulación transcripcional de los genes que codifican proteínas [34]. Recientemente, se mostró que cuatro de ellos pueden estar asociados con la tolerancia al etanol y con la respuesta al estrés [48]. Niederer y colaboradores mencionan en [34] que estudiar organismos modelo, que cuentan con una batería de herramientas genéticas, como lo es la levadura *S. cerevisiae* puede ayudar sustancialmente al gran desafió que representa la caracterización de los ARNs-lnc en otros organismos.

Por ello, buscando establecer una caracterización desde un punto de vista teórico, esta tesis se enfocó principalmente en hacer un análisis detallado de las gráficas de árbol con raíz asociadas a algunas secuencias de ARNs-lnc, de la levadura *S. cerevisiae*, con el fin de extraer información estructural. Esto permitió estudiar si existían algunas relaciones estructurales entre ellas y analizar si algunas estructuras provenientes de los ARNs-lnc estaban presentes, o no, en otros ARNs.

2.1.4. Secuencias de ARNs analizadas

En la sección 2.1.1 se estableció la importancia de estudiar los ARNs-lnc, mientras que en la sección 2.1.3 se abordó la importancia de la levadura *S. cerevisiae*. Así, siguiendo este camino, para ser analizadas se establecieron tres conjuntos de ARNs pertenecientes a la levadura *S. cerevisiae*. A saber:

• **Grupo control** $\bar{A}$: constó de las únicas 18 secuencias registradas en la literatura [46], hasta el año 2019, como ARNs-lnc.

• **Grupo control** $\bar{B}$: constó de 18 secuencias seleccionadas aleatoriamente de un conjunto de 64 ARNs que no pertenecen al conjunto de ARNs-lnc, ARNs-otros.

• **Grupo de prueba** $\bar{C}$: constó de seis secuencias de ARNs, los cuales fueron proporcio-

nados por la Dra. Lina Raquel Riego Ruiz y su equipo de la división de Biología Molecular, IPICYT; de las cuáles se sabía que tres de ellas no pertenecían al conjunto de ARNs-lnc, mientras que las tres restantes eran potencialmente ARNs-lnc.

El Cuadro 2.1 muestra los datos de las secuencias analizadas, así como el número de nucleótidos que las conforman (longitud, $|L|$).

Cuadro 2.1: ARNs que conforman los tres grupos analizados.

Grupo control $\bar{A}$		
ARN	SECUENCIA	$\|L\|$
ICR1	UCGAAAGACAUUCAUUCAAAAGAUUCUAACGUGAUUAUCGUAAAUAGCUGGA···	3199
RME2	CCCGCUGCAGUCUUUUUUAUGACCAUCUUUUUUAUGCAAUGCUAUUAAUGUU···	2223
RME3	UCGUUAUAUGAAAAGGCCCAUCUUGCAAUGAAAUAUUUCAUCCAAUCAAUAA···	1905
IRT1	UAGUUUAAAGAAUUUGAACUAUUUUUUGGCCAACUUGGAGAAAGAAUGUGUA···	1489
TLC1	AAUAAAACUAGAGAGGAAGAUAGGUACCCUAUGAAAAUGUCAAUGGCUGUUG···	1301
PWR1	CGGAUGUGUGGAUAAAAAGAACAUGGAAAACGAGUGCAGCAACGACUAUAUU···	941
RUF5-1	AACAAAGUAUCUAAACAAAAUACAUAAGUGUACUCAAACUGAGUAGAAUCGU···	710
RUF21	GUUUUCCAUUUUCUUGAAAUUAAAAAGAAAAAAAACCGAAAUAAUGAAAAAU···	707
ETS1-1	AUGCGAAAGCAGUUGAAGACAAGUUCGAAAAGAGUUUGGAAACGAAUUCGAG···	700
SRG1	UAAUGCCUUUGUUUGGCCAAGCUAUGUGCAAAUAUCACAAAUUAAAAAUUUG···	551
RUF22	GUUCAAUUAAUGAAUAAUUAGUUUGAUUAUUUUCCUCUUUAUUUGACCUUAG···	515
RUF20	ACGUGGGUUUUUUUUUCGAAUUGAGUGAUUAUGCAACCAUACAGGAACCUUA···	443
ITS1-1	AAGAAAUUUAAUAAUUUUGAAAAUGGAUUUUUUUGUUUUGGCAAGAGCAUGA···	361
RUF23	GCUAGGCAGAACGCCUAGUUUACACAGUGGGAGAAUGAGGAUAGGCCUCUGC···	254
ITS2-1	CCUUCUCAAACAUUCUGUUUGGUAGUGAGUGAUACUCUUUGGAGUUAACUUG···	232
ETS2-1	UUUUUAUUUCUUUCUAAGUGGGUACUGGCAGGAGCCGGGGCCUAGUUUAGAG···	211
RNA170	UUCUCUUAUAUACCAUGGCGCUGCAGAUCUAUCCAAAUUCUUCCAUACUGAA···	169
ZOD1	GCUUCUUGUUACCCCUUUUGGCGCUUUGGAUAAGUAGUUCACUUAGCUAUUU···	58

Grupo control $\bar{B}$		
ARN	SECUENCIA	$\|L\|$
15S Ribosomal	GUAAAAAAUUUAUAAGAAUAUGAUGUUGGUUCAGAUUAAGCGCUAAAUAAGG···	1649
LSR1	ACGAAUCUCUUUGCCUUUUGGCUUAGAUCAAGUGUAGUAUCUGUUCUUUUCA···	1175
Telomerase	GAGAGGAAGAUAGGUACCCUAUGAAAAUGUCAAUGGCUGUUGCGUUUGCUUA···	1158
SNR86	UAACCAUUGAUGAAAUUCUAUUGAAUGUCCCCAAUUUUCGGAAAAGGACUGC···	1004
SNR30	AACCAUAGUCUCGUGCUAGUUCGGUACUAUACAGGGAAGGGAAGUCACUCGC···	609
Small nuclear SNR30	AACCAUAGUCUCGUGCUAGUUCGGUACUAUACAGGGAAGGGAAGUCACUCGC···	606
SNR19	AUACUUACCUUAAGAUAUCAGAGGAGAUCAAGAAGUCCUACUGAUCAAACAU···	568
SNR84	AUUGCACAACUUAAGUUUGUCGAGGAUCAUUUUUUUGAACUGAAUCAUGCUC···	550
Small nuclear SNR84	AUUGCACAACUUAAGUUUGUCGAGGAUCAUUUUUUUGAACUGAAUCAUGCUC···	537
RPM1	AGAUAUUUAUAUUAUUAAUAGGAAAGUCAUAAAUAUAUAAAUUAUAUUAUAU···	483
SNR17B	GUCGACGUACUUCAGUAUGUUUUAUACCAUAUACUUUAUUAGGAAUAUAACA···	462
Nuclear RNASE P	ACAGUGGUAAUUCCUACGAUUAAGAAACCUGUUUACAGAAGGAUCCCCACCU···	358
SNR42	AUAACAAUUUGUGAUGCUUUAGGGAGCCUAUUGUUUGAUGGUUUAUAGGAGC···	351
NME1	AAUCCAUGACCAAAGAAUCGUCACAAAUCGAAGCUUACAAAAUGGAGUAAAA···	340
U3	UAACACAUUCUACAGUAGGAUCAUUUCUAUAGGAAUCGUCACUCUUUGACUC···	334
Small nuclear U3	UAACACAUUCAACAGUAGGAUCAUUUCUAUAGGAAUCGUCACUCUUUGACUC···	333
RNASE MRP	AUCCAUGACCAAAGAAUCGUCACAAAUCGAAGCUUACAAAAUGGAGUAAAAU···	332
SNR83	ACCCAAAAACAUCAAGAAAAGCCUUUCAAUAAAUUGCUCUUCUCUUGGCGAA···	306

Grupo de prueba $\bar{C}$		
ARN	SECUENCIA	$\|L\|$
KAP123	AUGGAUCAACAAUUUCUAAGUCAACUUGAGCAAACUUUGCACGCUAUCACUU···	3342
GRE2	AUGUCAGUUUUCGUUUCAGGUGCUAACGGGUUCAUUGCCCAACACAUUGUCG···	1029
ECM11	ATGACTGTTATAAAGACAGAACCAACAACAGAAGTGACATTATATTCTCCAC···	909
1477	AAAAAACCUACAUUUCAGAGUUAAAAUAAAUCUAAAGGUGCCACUGGAAAGA···	843
6754	GGAAACUACAAUAUAUUGCAAUAGAGCUUCCCCAAGGAACUCCCACUUUCCU···	840
12189	GGCCAGAAAUCAAAUUUAAAGGAAGUAGUGUACAAUAAGGUCGAUUUACUCC···	583

Si bien, el ARN es una molécula simple, aún se trata de entender las formas en que realiza muchas de sus funciones. Hasta ahora, se sabe que las modificaciones postranscripcionales y su estructura secundaria son de gran importancia en su función [18, 32, 38, 39, 40, 41]. Así, dado que la estructura secundaria del ARN es importante, en la siguiente sección se indagará un poco sobre ella.

2.2. Estructura secundaria del ARN

Hay que recordar que el ARN sigue una dirección en sentido $5'$ a $3'$ y está formado por ribosa, un fosfato y una de las cuatro bases nitrogenadas: adenina (A), citosina (C), guanina (G) y uracilo (U) [28]. Dado que el ARN es monocatenario, puede plegarse sobre sí mismo para alcanzar una estructura estable denominada estructura secundaria, la cual se forma mediante puentes de hidrógeno. Donde, la base A se complementa con la base U (A-U), mientras que la base C se complementa con la base G (C-G), a lo cual se le conoce como el apareamiento Watson y Crick [49], y en algunos casos la base G se complementa con la base U (G-U), a lo cual se le conoce como apareamiento por balanceo [50]. Así, la estructura secundaria, para el ARN, establece el apareamiento de bases: A-U, C-G y G-U.

Para poder generar la estructura secundaria de un ARN, es necesario contar con su secuencia FASTA que lo describe y utilizar un algoritmo de plegamiento, bajo ciertas condiciones estándares. Existen diversos programas para dicho fin, por ejemplo: programas basados en algoritmos de programación dinámica y cálculos termodinámicos, como: NUPACK Web [51], RNAfold WebServer [52] o MFold WebServer [53]. Programas que usan redes neuronales como rna-state-inf [54]. O incluso, programas que utilizan el aprendizaje profundo como MXfold2 [55]. Todos ellos, predicen el plegamiento del ARN por medio del apareamiento de bases, aunque por diferentes técnicas. Por ejemplo, en los primeros tres programas, las soluciones óptimas se obtienen minimizando la energía libre del ARN sobre la base de parámetros de energía libre, derivados experimentalmente para los pares de bases [56]. En el Ejemplo 3 se muestran las estructuras secundarias de los ARNs *ZOD1* y *RNA170*.

Ejemplo 3. *Usando los datos de la secuencia de los ARNs ZOD1 y RNA170, y el programa NUPACK Web, se puede asociar a dichos ARNs su estructura secundaria, las cuales pueden ser observadas en la Figura 2.2.*

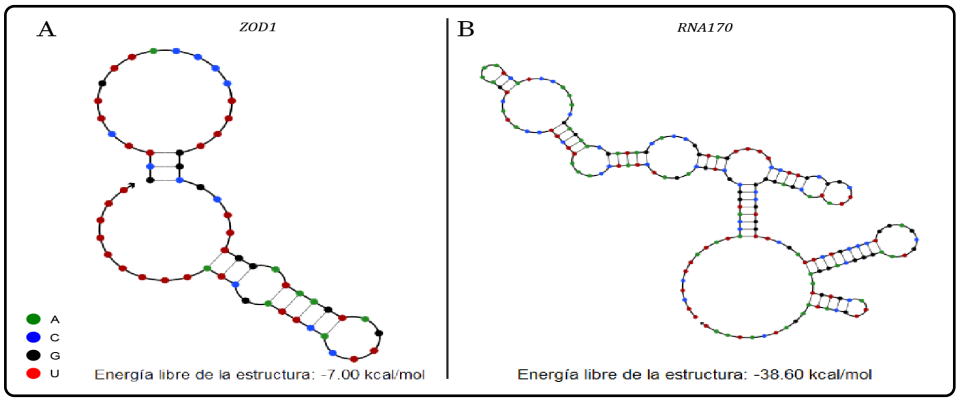


Figura 2.2: Dos estructuras secundarias de ARN generadas con NUPACK Web: en (A) *ZOD1* y en (B) *RNA170*.

Cabe remarcar que la estructura secundaria de un ARN no es única, dado que depende del programa de plegamiento que se utilice para generarla; sin embargo, comparten gran parte de su estructura. En el Ejemplo 4 se muestra una comparativa de la estructura secundaria de un mismo ARN pero generada con diferentes programas de plegamiento.

Ejemplo 4. *La Figura 2.3 muestra la estructura secundaria del RNA170 usando tres programas de plegamiento: NUPACK Web (en A), RNAfold WebServer (en B) y MFold WebServer (en C).*

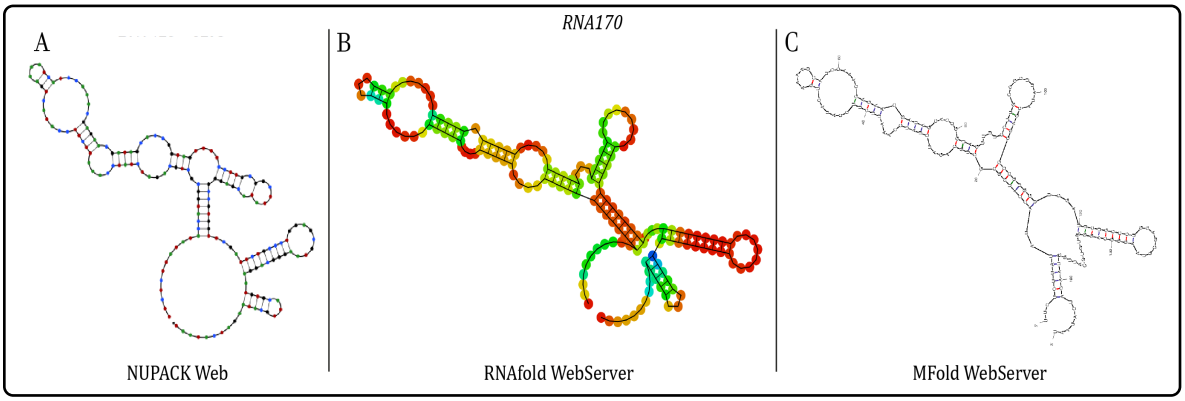


Figura 2.3: Estructura secundaria del *RNA170* generada con diferentes programas de plegamiento.

Después de generar varias estructuras secundarias, se puede notar que se presentan algunas formas particulares, a las cuales se les conoce como motivos estructurales. Los principales motivos estructurales que pueden estar presentes son: los tallos, las horquillas, las protuberancias y las uniones [57], los cuales están descritos en los incisos (i-iv) y se muestran en la Figura 2.4.

(i). Tallo: estructura que contiene dos o más pares de bases apareadas (A-U, C-G o G-U).

(ii). Horquilla: estructura en la cual una región no apareada permite que la secuencia se doble con el fin de formar un tallo.

(iii). Protuberancia: estructura que contiene una región no apareada en uno o en ambos lados de un tallo. La cual puede ser simétrica o asimétrica.

(iv). Unión: estructura donde tres o más tallos se unen.

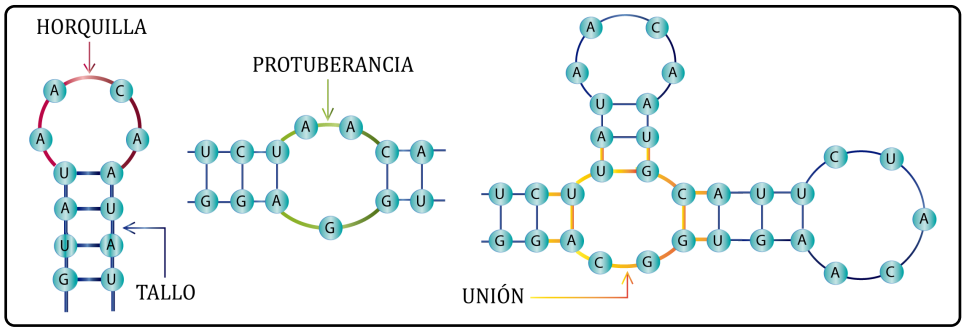


Figura 2.4: Motivos estructurales que pueden estar presentes en la estructura secundaria del ARN.

Así, usando los datos de una secuencia de ARN se le puede generar su correspondiente estructura secundaria. En esta tesis se usó como base el programa NUPACK Web. Ahora bien, después de analizar varias estructuras secundarias surge una pregunta de manera natural, ¿qué tanto se parecen (o difieren) las estructuras? De manera general, es fácil determinar si dos estructuras son iguales. Pero ¿si no lo son? Una aproximación para buscar establecer qué tan parecidas son las estructuras es determinar si existen, o no, similitudes entre ellas. La idea de buscar similitudes en un principio es sencilla. Pero ¿qué pasa cuando las estructuras secundarias para comparar no son tan simples?, como por ejemplo, las estructuras secundarias de los ARNs *TLC1* y *RUF21*, de longitudes: 1301 y 707 nucleótidos, respectivamente, mostradas en el Ejemplo 5.

Ejemplo 5. *En la Figura 5 se muestran las estructuras secundarias de los ARNs TLC1 y RUF21.*

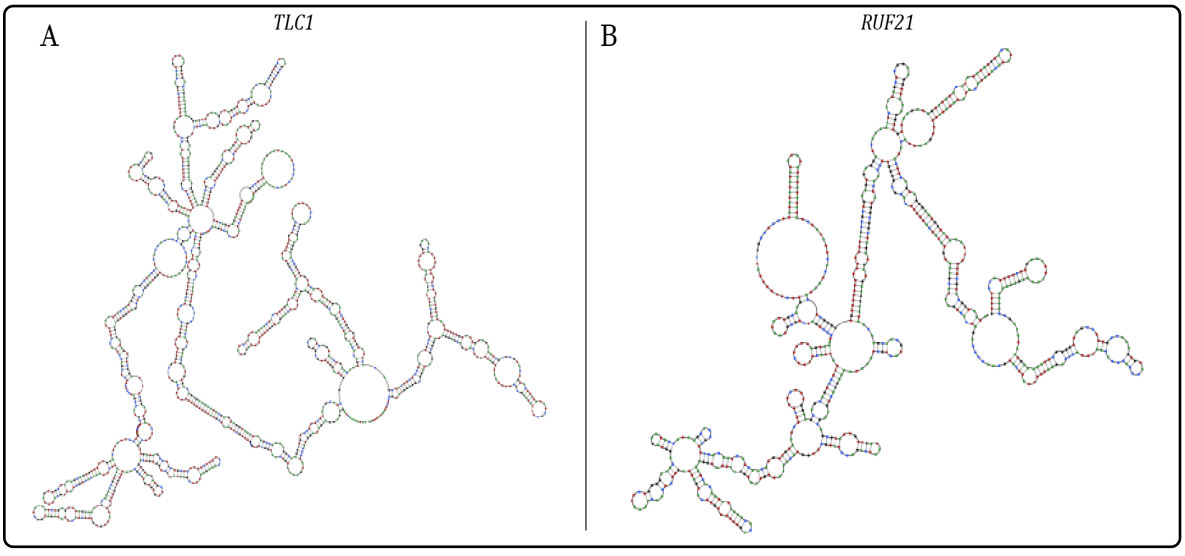


Figura 2.5: Las estructuras secundarias de los ARNs *TLC1* (en A) y *RUF21* (en B).

Dado que es importante estudiar las estructuras secundarias asociadas a los ARNs, especialmente a los ARNs-lnc, y establecer qué tan parecidas son, en el Capítulo 3 se establecerá un análisis basado en gráficas de árbol con raíz para dicho fin. Cabe señalar que también se exploró el análisis de la estructura primaria del ARN; sin embargo, con dicho análisis no se pudo establecer una caracterización que permitiera identificar a los ARNs-lnc de los que no lo son. En el Apéndice B se encuentra un breve resumen sobre dicho análisis.

Capítulo 3

ARNs y gráficas

El estudio del ARN ha proporcionado un nuevo campo de oportunidad, especialmente para nuevos enfoques conceptuales y matemáticos. De hecho, las matemáticas y la informática en los últimos años han proporcionado herramientas prometedoras para la investigación de la biología estructural [11, 57]; en particular, para explorar el repertorio de las estructuras secundarias del ARN [4, 58]. Gan *et al.* [4] establecieron un marco teórico-gráfico para representar una secuencia de ARN mediante una gráfica de árbol, la cual se tomará como punto de partida en el desarrollo de esta tesis. Así, es importante establecer algunos conceptos de la teoría de gráficas.

La teoría de gráficas es una rama de las matemáticas que estudia las configuraciones descritas mediante vértices y aristas, a las cuales se les llama gráficas. La representación geométrica de una gráfica está formada por puntos (vértices) y por líneas (aristas) que establecen cómo se unen dichos puntos. Para entender un poco sobre gráficas y sus propiedades se emplearán las definiciones establecidas por Diestel en [59]. Así, una gráfica $\mathcal{G}$ será un par ordenado $\mathcal{G} = (V, E)$, donde $E \subset V \times V$. Un árbol será una gráfica conectada sin ciclos, cómo la del Ejemplo 6. La gráfica de árbol puede ser dibujada en el plano de tal manera que ninguna de sus aristas se cruce. Cabe señalar que en esta tesis se analizaron gráficas de árbol planas; es decir, gráficas contenidas en el plano $\mathbb{R}^2$.

Ejemplo 6. *Dado* $V = \{A, B, C, D, E\}$ *y* $E = \{(A,B), (B,C), (B,D), (B,E)\}$, *la Figura 3.1 muestra su respectiva gráfica de árbol.*

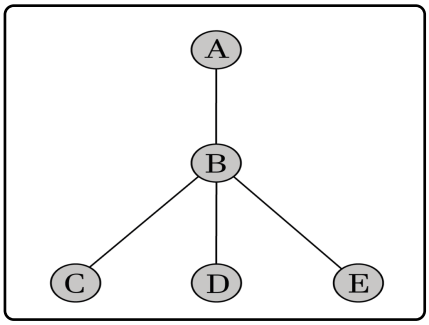


Figura 3.1: Ejemplo de una gráfica de árbol.

Un árbol (V,E) con un vértice seleccionado $r \in V$ es un árbol con raíz, denotado por (V,E,r), donde el vértice r es llamado la raíz del árbol. Algunas veces, el nombre de los vértices no se muestran en la gráfica y la raíz del árbol se resalta, cómo se muestra en la Figura 3.2. Cabe señalar que un isomorfismo entre árboles con raíz preserva la raíz.

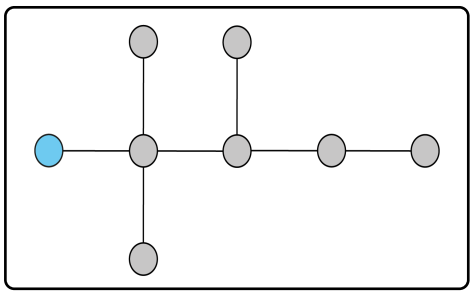


Figura 3.2: Esquema de un árbol con raíz. La raíz del árbol está resaltada en color azul.

Ahora bien, con lo anterior en mente se definirá una herramienta básica que será de utilidad, llamada la Notación Punto-Paréntesis Simplificado (*NPPS*), la cual puede ser asignada a una gráfica de árbol con raíz.

La *NPPS* asigna a una gráfica de árbol con raíz una cadena descrita por puntos y paréntesis, mediante las siguientes reglas: empezando desde la raíz de la gráfica de árbol (y recorriendo la hasta el final), se seguirá el contorno de la gráfica a favor de las manecillas del reloj. Esto quiere decir que se pasará por cada arista dos veces y por cada vértice el mismo número de aristas que lo generan, a excepción del vértice raíz que solo se pasará por él dos veces. Luego, siguiendo dicho recorrido, se le asignará un punto cada que se encuentre un vértice, •, a excepción del vértice raíz que posee un punto extra. Además, la primera vez que se encuentre con una arista se le asignará un paréntesis abierto, (, mientras que si es la segunda vez que se pasa por la misma arista se le asignará un paréntesis cerrado,). En el Ejemplo 7 se le asociará a una gráfica de árbol con raíz su *NPPS*.

Ejemplo 7. *La NPPS asociada a la gráfica de árbol con raíz mostrada en la Figura 3.2 es* .(.(.).(.(.).(.(.).).).(.).). *, mientras que, la NPPS asociada a la gráfica de árbol con raíz* (V,E,A) *mostrada en la Figura 3.1 es* .(.(.).(.).(.).).

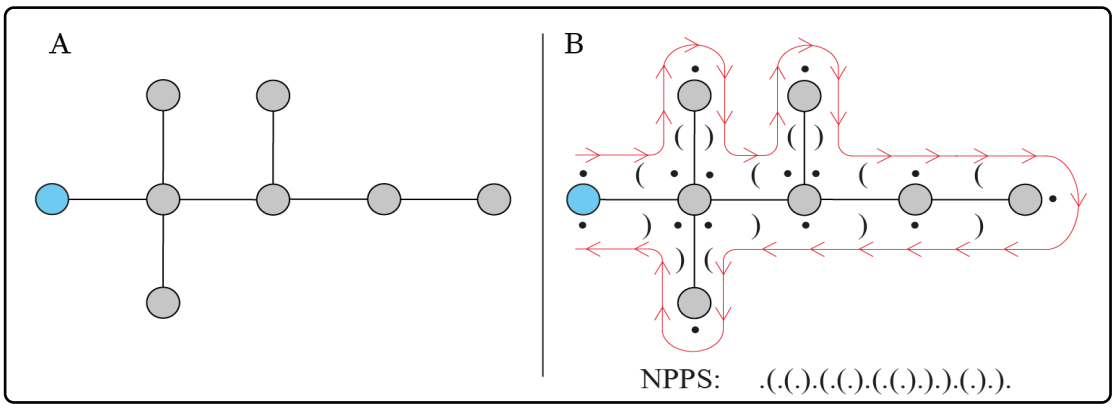


Figura 3.3: La gráfica de árbol con raíz de la Figura 3.2 (en A), y siguiendo las flechas rojas a favor de las manecillas del reloj y asociándole: •, (ó), de acuerdo a las reglas anteriores, se obtiene su *NPPS* asociada: .(.(.).(.(.).(.(.).).).(.).). (en B).

En la Figura 3.4 se muestran dos gráficas de árbol con raíz y sus correspondientes *NPPS*. Notar que, a pesar de que las gráficas son muy similares, es posible notar la diferencia entre sus cadenas en *NPPS*.

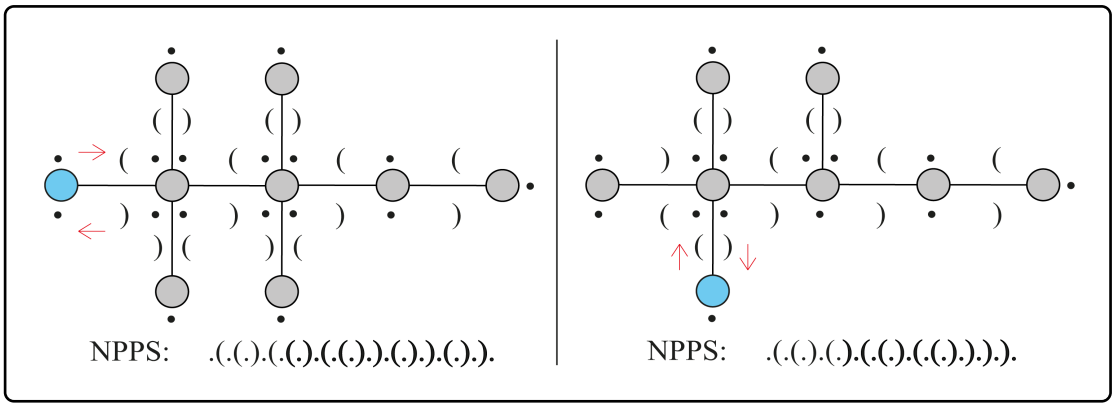


Figura 3.4: Dos gráficas de árbol con raíz y sus cadenas en *NPPS* correspondientes, donde la diferencia entre sus cadenas en *NPPS* están remarcadas.

En las siguientes secciones se mostrará que el usar una cadena en *NPPS* es útil para estudiar gráficas asociadas a la estructura secundaria proveniente de una secuencia de ARN.

3.1. De estructura secundaria a Notación Punto-Paréntesis

Dada una secuencia de ARN y su estructura secundaria, es común asociarle una cadena en Notación Punto-Paréntesis (*NPP*). La *NPP* es una cadena de texto compuesta por puntos y paréntesis, la cual contiene la información de cómo está conformada la estructura secundaria del ARN en cuestión. Cabe remarcar que, cada carácter de dicha cadena representa la interacción de cada nucleótido. A saber, un paréntesis abierto, (, indica que dicho nucleótido esta apareado con otro delante de él, mientras que, un paréntesis cerrado,), indica que dicho nucleótido está apareado con otro anterior a él, finalmente, un punto, •, indica que dicho nucleótido no está apareado con algún otro. La secuencia $E1$ y su *NPP* asociada se pueden consultar en el Ejemplo 8. Cabe señalar que la secuencia $E1$ no corresponde a una secuencia de ARN. Sin embargo, será de utilidad para remarcar algunas de las características descritas por las reglas de Gan.

Ejemplo 8. *Datos de la secuencia* $E1$*:*

ACCCGGCCACAGUGAGCGGAACACCCGUGACUCAUUUCGAACCUCGGAAGUUAAGCCG
CUCACGUUGGUGGGGCCGUGGAUAACCGUGAGGAUCCGCAGCCCCACUAAGCUGGGAU

Su estructura secundaria es mostrada en la Figura 3.5, donde las horquillas, protuberancia y uniones, así como el inicio y final de su secuencia ($5'$ *y* $3'$*) están numeradas. Por otro lado, la NPP que describe a dicha secuencia esta descrita por:*

.((((((....((((((((((..(((.(..((.....))..).)))..)))...)))))))).(((((((((.((((((..((....)))))))).)))))))))))..

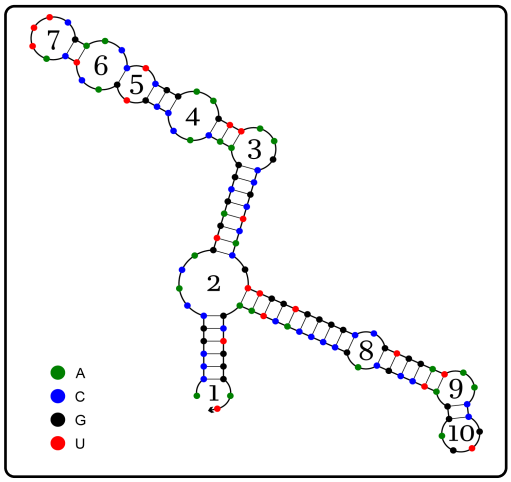


Figura 3.5: Estructura secundaria correspondiente a los datos de la secuencia $E1$.

Cabe remarcar que desde el punto de vista topológico, son de interés las formas presentes en la estructura secundaria del ARN; es decir, sus tallos, bucles, horquillas y uniones, más que el número de nucleótidos que conforman a cada motivo estructural. Por ello, resulta útil asociar una gráfica de árbol con raíz a una estructura secundaria de ARN, como veremos en la siguiente sección.

3.2. La gráfica de árbol con raíz asociada

En la sección 2.2 a una secuencia de ARN se le generó su correspondiente estructura secundaria. Ahora, buscando analizar dichas estructuras desde el enfoque de la teoría de gráficas, un camino congruente es emplear las gráficas de árbol con raíz. Para ello, a una estructura secundaria de ARN se le asociará una gráfica de árbol con raíz siguiendo el enfoque teórico-gráfico propuesto por Gan y colaboradores en [4].

Reglas de Gan:

(i) A los extremos $5'$ y $3'$ del tallo se les considerará como el vértice raíz.

(ii) A una protuberancia o a una horquilla, se le considerará como un vértice cuando haya dos o más nucleótidos consecutivos no apareados.

(iii) A una unión se le considerará como un vértice.

(iv) A un tallo se le considerará como una arista si este posee al menos dos pares de bases complementarias.

Cabe señalar que los extremos $5'$ y $3'$ se consideran como un solo vértice dado que en una gráfica de árbol cualquier arista debe unir dos vértices. Por otro lado, los demás vértices podrían participar en la formación de la estructura del ARN con bases no apareadas

en otras partes de la molécula mediante interacciones terciarias, lo que permite estabilizar la estructura tridimensional del ARN y, por lo general, implica que dichas estructuras posean más de un par de bases complementarias. Por lo tanto, las protuberancias, los bucles y las uniones representadas como vértices, son determinantes en la interacción, la flexibilidad y la estructura terciaria del ARN. Asimismo, un mínimo de dos pares de bases complementarias garantiza la estabilidad del tallo frente a las fluctuaciones térmicas.

Para entender mejor las reglas de Gan, en el Ejemplo 9 se mostrará cómo se aplican a una estructura secundaria en particular.

Ejemplo 9. *Usando como base la estructura secundaria mostrada en la Figura 3.6 (A), correspondiente a la estructura secundaria de la secuencia* $E1$. *Se usarán las reglas de Gan para asignarle su gráfica de árbol con raíz mostrada en la Figura 3.6 (B).*

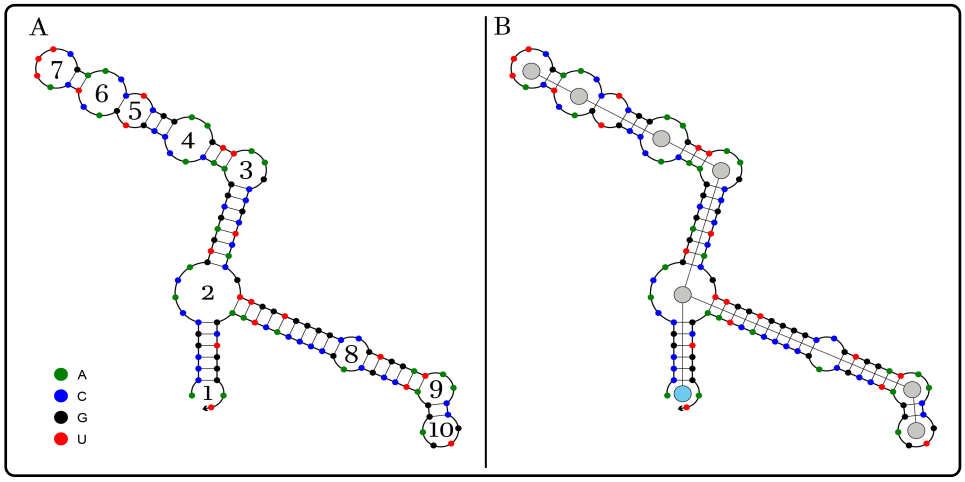


Figura 3.6: La estructura secundaria de la secuencia $E1$ (en A) y su gráfica de árbol con raíz asociada (en B).

Para asociarle a la estructura secundaria mostrada en la Figura 3.6 (A) su gráfica de árbol con raíz (B), se debe: asignar un vértice raíz al extremo $5'$ *y* $3'$ (1), *regla (i). Luego, notar que la protuberancia* 8 *no satisface la regla (ii), por lo tanto, no se le asigna un vértice, mientras que a todas las demás protuberancias* $(3,4,9)$, *uniones* (2) *y horquillas* $(7,10)$ *se les asigna un vértice, regla (ii) y (iii). Después, dado que el tallo entre la protuberancia* 5 *y* 6 *tiene solo un par de bases complementarias, ambas protuberancias se consideran como una sola y se les asigna un vértice, regla (iii), mientras que a los demás tallos se les asigna una arista, regla (iv).*

De manera análoga, Gan *et al.* [4] establecen también una serie de reglas para poder asignarle a una gráfica de árbol o a una estructura secundaria de ARN que contenga pseudonudos, su gráfica plana dual. Un pseudonudo [60] es una estructura de ARN compuesta por al menos dos segmentos helicoidales conectados por regiones monocatenarias o bucles, generalmente están presentes cuando se analiza la estructura terciaria del ARN. Si bien es cierto que las gráficas duales son útiles para analizar la estructura terciaria del ARN y en los

casos especiales donde surgen pseudonudos, en esta tesis no se emplearán las gráficas duales ya que, por un lado, el análisis está enfocado en la estructura secundaria del ARN. Y por el otro lado, por defecto, la mayoría de los programas de plegamiento excluyen la formación de pseudonudos en los plegamientos de la estructura secundaria. Con ello en mente, solo se utilizarán las gráficas de árbol con raíz.

Así, dada una estructura secundaria de ARN se le ha asociado una gráfica de árbol con raíz mediante las reglas descritas por Gan [4]. A primera vista es un proceso sencillo, pero una vez más depende del nivel de interpretación para poder hacer cada asignación, proceso que se complica cuando la estructura secundaria que se analiza no es sencilla, como lo es la estructura secundaria mostrada en el Ejemplo 10.

Ejemplo 10. *A pesar de que la estructura secundaria del ARN RUF23 no es tan grande, asociarle su gráfica de árbol con raíz ya no es una tarea trivial. Su estructura secundaria y su gráfica de árbol con raíz se muestran en la Figura 3.7.*

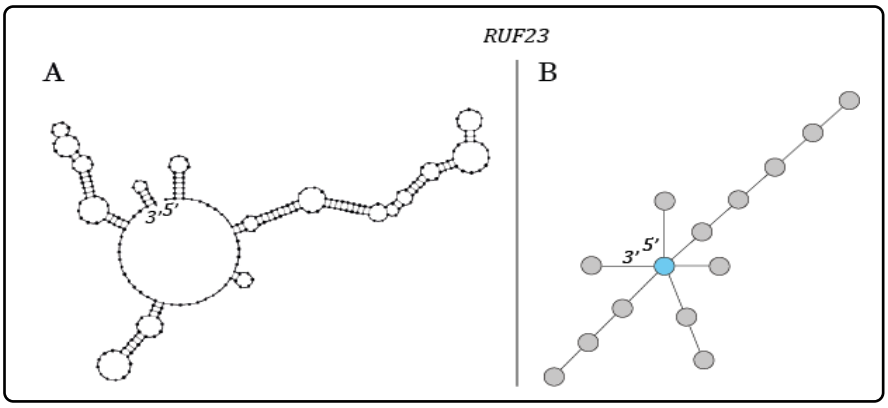


Figura 3.7: La estructura secundaria del ARN *RUF23* (en A) y su gráfica de árbol con raíz asociada (en B).

Dado que es importante el analizar las gráficas de árbol con raíz provenientes de estructuras secundarias de ARNs y dada la complejidad de emplear las reglas establecidas por Gan [4] a secuencias de ARN de grandes longitudes, se buscará establecer una metodología que no dependa del nivel de interpretación para hacer las asignaciones correspondientes.

Resumiendo hasta este punto, por un lado se tiene que a una gráfica de árbol con raíz se le puede asignar su *NPPS*. Por otro lado, una estructura secundaria de ARN tiene una *NPP* asociada y a dicha estructura se le puede asignar una gráfica de árbol con raíz, el Ejemplo 11 muestra una estructura secundaria, su *NPP*, su gráfica de árbol con raíz y su *NPPS*.

Ejemplo 11. *La estructura secundaria mostrada en la Figura 3.8 (A) tiene la siguiente NPP asociada:*

.((((((....((((((((((..(((.(..((.....))..).)))..)))...)))))))).((((((((((.((((((..((....))))))).))))))))))))..

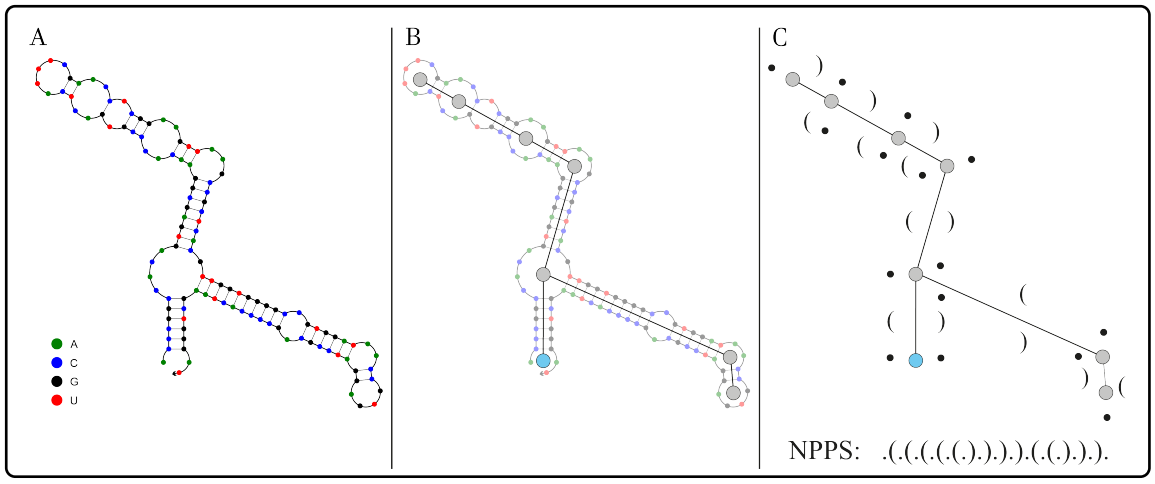


Figura 3.8: La estructura secundaria de la secuencia $E1$ (en A), su gráfica de árbol con raíz asociada mediante las reglas de Gan (en B) y su gráfica de árbol con raíz con su *NPPS* asociada (en C).

Cabe señalar que la *NPP* del Ejemplo 11 codifica la estructura secundaria de la secuencia $E1$ mostrada en la Figura 3.8 (A), mientras que la *NPPS* mostrada en la Figura 3.8 (C) codifica la gráfica de árbol con raíz mostrada en la Figura 3.8 (B) asociada a la estructura secundaria mostrada en la Figura 3.8 (A). Una tarea interesante es obtener la *NPPS* solo de la *NPP*. Así, dada una *NPP* la meta es aplicar las reglas de Gan y obtener la *NPPS* que codifique la gráfica de árbol con raíz asociada.

Vale la pena notar que, de manera natural, el reducir en la *NPP* los mismos caracteres a uno solo que los represente, parece apropiado para obtener la *NPPS*. Pero, por ejemplo, en el caso de la *NPP* mostrada en el Ejemplo 11:

.((((((....((((((((((..(((.(..((.....)).).)))..)))...)))))))).(((((((((.((((((..((....)))))))).)))))))))))).. ,

se tiene que al reducir los mismos caracteres a uno solo que los represente se genera como resultado la cadena:

.(.(.(.(.(.).).).).).(.(.(.).). ,

la cual no esta asociada a ninguna gráfica de árbol con raíz, dado que en dicha cadena el número de paréntesis abiertos es diferente al de los paréntesis cerrados. Sin embargo, como se mostró en la Figura 3.8 (C) la correcta asociación de su *NPPS* es .(.(.(.(.(.).).).).(.(.).).). Así, buscando establecer una correcta asociación, en la siguiente sección se expondrá una metodología que permitirá asociarle a la *NPP* la *NPPS*.

3.3. De una Notación Punto-Paréntesis a una simplificada

Hasta este punto, dada una secuencia de ARN descrita en formato FASTA, se puede usar un programa de plegamiento para obtener tanto su estructura secundaria como su *NPP* que la describe, para esta tesis se utilizó NUPACK Web, podrá encontrar un breve manual de cómo usar este programa en al Apéndice C. Ahora bien, siguiendo las reglas de Gan se

establecerá un procedimiento que permitirá asignar a la notación punto-paréntesis, *NPP*, la notación punto-paréntesis simplificada, *NPPS*, asociada a su gráfica de árbol con raíz. Para poder obtener la *NPPS* se deberá modificar la estructura secundaria original en *NPP* bajo el siguiente procedimiento:

I. Si un tallo consiste solo de un par de bases complementarias, este debe ser removido. Además, las protuberancias y horquillas que no tengan más de dos nucleótidos consecutivos no apareados también deben ser removidos.

II. Cada protuberancia y cada unión debe tener al menos un punto que la separe de los tallos convergentes a ella, si no es el caso, se debe insertar un punto.

III. Por último, los mismos caracteres consecutivos deben ser reducidos a uno solo que los represente.

Ahora bien, siguiendo el procedimiento anterior y señalando de un paso a otro los cambios realizados, en el Ejemplo 12 a la *NPP* de la estructura secundaria mostrada en la Figura 3.9 (A) se le asociará su *NPPS*.

Ejemplo 12. *Tomando como referencia la Figura 3.9 (A), se partirá de su NPP y se obtendrá su NPPS. Para ello, en (a) se muestra su NPP correspondiente a dicha estructura secundaria. Notar que, dado a que el tallo entre la protuberancia* 5 *y* 6 *solo tiene una base complementaria, esta debe ser removida, obteniendo así (b). Dado que la protuberancia* 8 *no tiene más de dos nucleótidos consecutivos no apareados este debe ser removido, obteniendo (c). En las protuberancias* 2, 3 *y* 9 *se debe insertar un punto en las regiones marcadas con las letras:* x, y, z*, como se muestra en la Figura 3.9 (B), obteniendo con ello (d). Finalmente, reducir los mismos caracteres consecutivos a uno solo que los represente para obtener (e). Cabe señalar que la secuencia mostrada en (e) es la NPPS de la gráfica de árbol asociada a dicha estructura secundaria.*

(a) .((((((....((((((((((..(((.(..((.....)).).))).)))...)))))))).(((((((((.((((((..((....)))))))).))))))))))..

(b) .((((((....((((((((((..(((.•..((.....)).•.))).)))...)))))))).(((((((((.((((((..((....)))))))).))))))))))..

(c) .((((((....((((((((((..(((....((.....)).....))).)))...)))))))).(((((((((**(**(((((((..((....)))))))))**)**))))))))))..

(d) .((((((....(((((((•(((..(((....((.....)).....))).)))...)))))))).((((((((((((((((..((....))•))))))))))))))))•))))))..

(e) .(.(.(.(.(.(.).).).).(.(.).).).

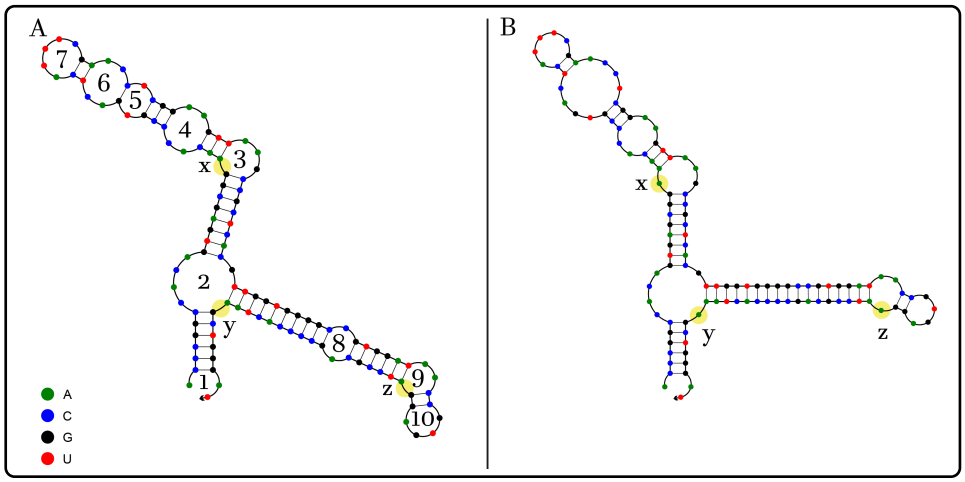


Figura 3.9: La estructura secundaria de la secuencia $E1$ obtenida de (a) y en amarillo se muestran las regiones x, y, z donde un punto debe ser insertado para obtener su *NPPS* (en A), y su estructura secundaria obtenida de (d) y en amarillo remarcadas las mismas regiones donde el punto fue insertado (en B).

Cabe señalar que con este procedimiento, diferentes estructuras secundarias de ARN podrían tener asociadas la misma *NPPS* y, por lo tanto, la misma gráfica de árbol con raíz asociada. Esto, dado a que este análisis se enfoca en la gráfica de árbol con raíz más que en la longitud de los tallos, protuberancias o uniones que están presentes en su estructura secundaria.

En el Capítulo 4, dados dos conjuntos con características distintas cada uno, se buscará analizar y determinar si existen similitudes estructurales entre ellos. Así, en la sección 4.1, se describe un algoritmo para llevar a cabo dicha tarea. A partir de este punto, nos referiremos a las gráficas de árbol con raíz sólo como gráficas.

Capítulo 4

Algoritmo de comparación

Dados dos conjuntos con distintas características cada uno, se busca establecer si existen similitudes estructurales entre ellos o no. Con el fin de realizar dicho análisis, se conformaron tres conjuntos de ARNs, utilizando para ello sus secuencias en formato FASTA. Los primeros dos grupos serán los grupos control $\bar{A}$ y $\bar{B}$, de los cuales es de interés saber si sus elementos tienen alguna característica estructural que juegue un papel importante en su función. El tercer grupo será el grupo de prueba $\bar{C}$, en el cual se buscará si sus elementos tienen la característica estructural determinada. A saber, en esta tesis la característica funcional de interés es el ser un ARN-lnc. Por ello, los elementos del conjunto $\bar{A}$ son ARNs-lnc validados, mientras que los del conjunto $\bar{B}$ son elementos que no son ARNs-lnc. Finalmente, los elementos del grupo de prueba $\bar{C}$ son los ARNs en los que se analizará si tienen dicha característica funcional;es decir, la posibilidad de ser ARNs-lnc.

Recordar que, para los datos de las secuencias de ARN, se requiere generar su estructura secundaria y su *NPP* asociada. En esta tesis se empleó NUPACK Web para dicho fin. Así, la información que tiene cada conjunto, $\bar{A}$, $\bar{B}$ y $\bar{C}$, es: el nombre de los ARNs que serán analizados, su secuencia de nucleótidos descrita en el formato FASTA y su cadena en *NPP* asociada. Las secuencias serán ordenadas de mayor a menor, tomando como referencia el número de nucleótidos que la conforman. Además, se requiere obtener sus *NPPS*; para ello, se usaran sus *NPP* y el procedimiento establecido en la sección 3.3. Finalmente, se obtendrán sus respectivos conjuntos A, B y C descritos en *NPPS*. Dicho procedimiento está esquematizado en la Figura 4.1.

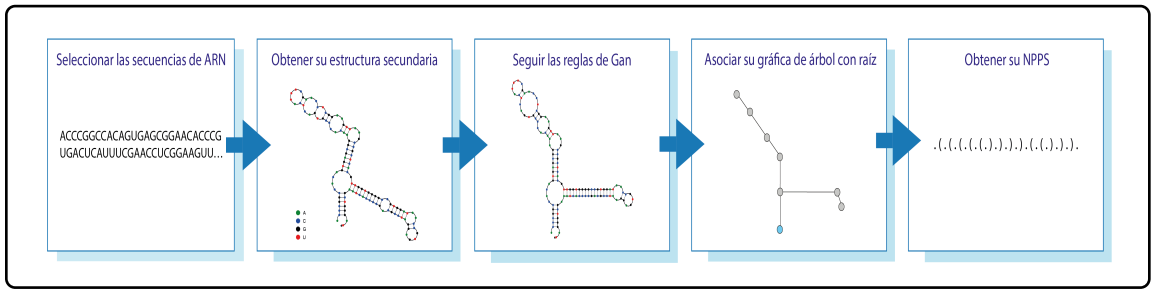


Figura 4.1: Procedimiento para obtener la *NPPS* de una secuencia de ARN.

Ahora bien, dado los conjuntos A, B y C, descritos en la *NPPS*, el algoritmo propuesto pretende analizar dichos conjuntos y buscar entre ellos si existen o no cadenas repetidas. La idea principal del algoritmo es, determinar las cadenas compartidas que aparecen con mayor frecuencia dentro de los conjuntos A y B y determinar si los elementos en el conjunto C comparten más cadenas repetidas con el conjunto A, o con el conjunto B. Así, dicho elemento será más probable de tener la misma funcionalidad que el conjunto correspondiente, dicho procedimiento esta esquematizado en la Figura 4.2. Este algoritmo fue implementado en Python v.3.8, en el Apéndice E está su pseudocódigo correspondiente.

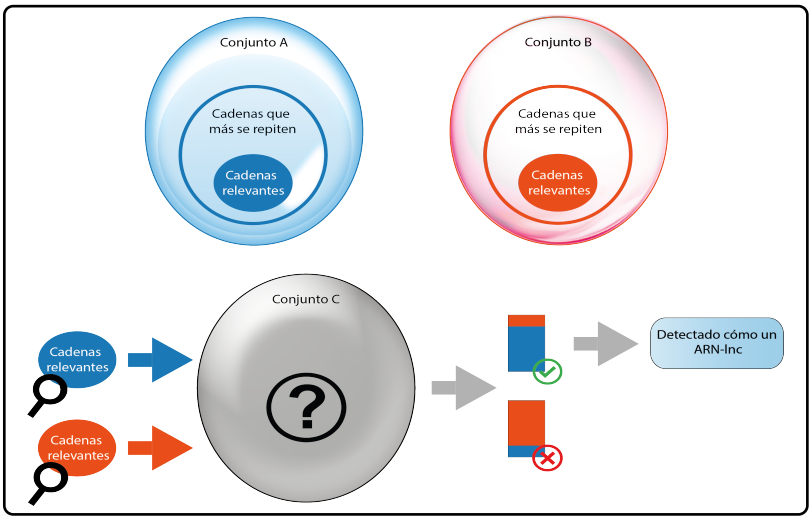


Figura 4.2: Un esbozo del algoritmo propuesto. Se identifican las cadenas relevantes en los conjuntos A y B, las cuales se buscan si están presentes en el conjunto C. Un elemento del conjunto C será detectado como un probable ARN-lnc cuando su cadena esté conformada por más cadenas relevantes del conjunto A y menos cadenas relevantes del conjunto B.

En la sección 4.1, se dará en bosquejo del algoritmo, que dados los conjuntos A, B y C, descritos en *NPPS*, comparará las cadenas compartidas y examinará la posibilidad que tienen los elementos del conjunto C en tener la característica estructural determinada.

4.1. El algoritmo

Como se mencionó previamente, dados los conjuntos $\bar{A}$, $\bar{B}$ y $\bar{C}$, con los datos de las secuencias de ARN, se utiliza un programa de plegamiento para generar su estructura secundaria en *NPP*, $\hat{A}$, $\hat{B}$ y $\hat{C}$, respectivamente. Después, se utilizan las reglas de Gan para obtener los conjuntos en *NPP*, denotados por A_G, B_G, y C_G, respectivamente, buscando establecer sus gráficas de raíz asociadas. Finalmente, empleando el método establecido en la sección 3.3, se simplifica su *NPP* a su *NPPS*, la cual codifica la información importante de la gráfica asociada, obteniendo con ello los conjuntos A, B, y C, respectivamente. Cabe remarcar que el algoritmo propuesto emplea, como datos de entrada, conjuntos de secuencias de ARN en *NPPS*, en este caso los conjuntos A, B, y C.

Ahora, dado dos parámetros enteros l_1 y l_2, se buscan en el conjunto A (y en B) las cadenas que se repitan de longitud l que satisfagan la desigualdad $l_1 \leq l \leq l_2$, para con ello establecer el conjunto de cadenas repetidas S_A (y S_B). Cabe señalar que la elección de los

parámetros l_1 y l_2 depende de los conjuntos involucrados. En el análisis de estos conjuntos en particular se sugiere que $l_1 \geq 18$, dado que longitudes más cortas no aportan información relevante y ocasionan que el análisis sea más lento e incrementa la probabilidad de obtener falsos positivos. Mientras que se siguiere que $l_2 \leq 34$, dado que una cadena de longitud mayor es menos probable que se repita en otras cadenas.

Primero, para determinar cuáles de las cadenas en S_A son más relevantes para detectar ARNs-lnc, se buscan las cadenas del conjunto S_A en A y en B para con ello obtener las matrices M_{AA} y M_{AB}, respectivamente. En las cuales, la ij-ésima entrada de la matriz M_{AA} (o M_{AB}) es el número de veces que la j-ésima cadena de S_A aparece en el i-ésimo elemento de A (o B). Notar que si A y S_A tienen m y n elementos, respectivamente, entonces, M_{AA} es una matriz de $m \times n$. Análogamente, se realiza el procedimiento para el conjunto S_B para con ello obtener las matrices M_{BA} y M_{BB}.

Segundo, se comparan las matrices M_{AA} y M_{AB} para con ello determinar qué cadenas en S_A pertenecen al conjunto de cadenas relevantes de A, conjunto $S_A^{'}$. Este conjunto será importante para detectar ARNs-lnc. Para este fin, se suman las columnas en la matriz M_{AA} (y en M_{AB}) para obtener con ello el vector C_{AA} (y C_{AB}) y se establece un discriminante adecuado l_3 con el fin de evaluar $C_{AA} - C_{AB} \geq l_3$. Donde, $C_{AAi} - C_{ABi} \geq l_3$ significa que el i-ésimo elemento de S_A pertenece al conjunto $S_A^{'}$, cabe notar que si $C_{AAi} - C_{ABi} = 0$, dicho elemento aparece el mismo número de veces en A y en B, así dicho elemento no es importante para identificar ARNs-lnc. De igual forma, se comparan las matrices M_{BA} y M_{BB} para calcular $C_{BB} - C_{BA} \geq l_3$ y así determinar el conjunto de cadenas relevantes de B, $S_B^{'}$.

Tercero, se define el conjunto $D = A + B + C$. De manera análoga al procedimiento anterior, se buscan los elementos de $S_A^{'}$ en A y en D, obteniendo con ello las matrices $M_{AA}^{'}$ y $M_{AD}^{'}$, respectivamente. De igual forma, se buscan los elementos de $S_B^{'}$ en B y en D obteniendo con ello las matrices $M_{BB}^{'}$ y $M_{BD}^{'}$. Ahora, para $M_{AA}^{'}$ se calcula la suma de la fila del vector R_{AA}, nótese que cada R_{AAi} indica el número de cadenas relevantes en el i-ésimo elemento, de manera similar se calculan los vectores R_{BB}, R_{AD} y R_{BD}.

Cuarto, se calcula el primer cuartil[1] $Q1_A$ (y $Q1_B$) del vector R_{AA} (y R_{BB}), los cuales se emplearán para establecer un rango que asegure una mayor probabilidad para que un ARN sea un ARN-lnc. La hipótesis que se estableció fue la siguiente: si un ARN posee muchos elementos de $S_A^{'}$ y pocos elementos de $S_B^{'}$ es más probable que dicho ARN pertenezca al conjunto A. Finalmente, se determina cuales de las i-ésimas entradas de R_{ADi} en R_{AD} y cuales de las i-ésimas entradas de R_{BDi} en R_{BD} satisfacen la siguiente restricción:

$$R_{ADi} > 0.5 * Q1_A \qquad \text{y} \qquad R_{BDi} < 1.5 * Q1_B.$$

Esto quiere decir que, el i-ésimo elemento de A posee muchas de las cadenas relevantes del conjunto $S_A^{'}$ y pocas cadenas relevantes del conjunto $S_B^{'}$. Así, en este caso, el i-ésimo elemento en D que satisface dicha restricción es detectado como un ARN-lnc. Por lo tanto, se puede asignar un vector V_X de longitud $|X|$, donde $X \in \{A, B, C, D\}$ de la siguiente manera:

[1]Los cuartiles son una medida estadística de posición, que dividen un conjunto de datos en cuatro partes iguales. Donde el primer cuartil $Q1$ muestra cómo el 25% de los datos están agrupados, de igual forma: $Q2$ el 50% y $Q3$ el 75% . Buscando con ello, mostrar la dispersión de los elementos del conjunto.

si el i-ésimo elemento en X es detectado como un ARN-lnc, se le asignará un uno a la i-ésima posición del vector V_X; mientras que en otro caso, se le asignará un cero. Así, dados los conjuntos A, B, y C, el vector resultante mediante el algoritmo es: $V_D = V_A + V_B + V_C$.

Claramente, más elementos de A deben ser detectados como ARNs-lnc que los detectados en B. Así, bajo esta hipótesis, los elementos en C también detectados tienen mayor probabilidad de ser ARNs-lnc. Es cierto que, con algunas combinaciones de conjuntos A, B y C, esto no se satisface. En este sentido, se estableció una restricción que permite seleccionar los casos en los cuales más elementos de A son detectados más que los de B. Así, dado los números N_A y N_B correspondientes a A y B, respectivamente, un elemento de C es más probable de ser un ARN-lnc cuando se satisface que: $N_A \geq l_4$ y $N_B \leq l_5$. Aquí, l_4 y l_5 se sugieren que satisfagan: $l_4 \geq \frac{|A|}{3}$ y $l_5 \leq \frac{|B|}{4}$, donde $|A|$ y $|B|$ representan la cardinalidad de A y B, respectivamente.

En conclusión, dados los conjuntos A, B y C el algoritmo, que denotaremos como:

$$\mathcal{A}(A,B,C,l_1,l_2,l_3,l_4,l_5)$$

O simplemente $\mathcal{A}(A,B,C)$, muestra como resultado un vector V_C. Por ejemplo, si el conjunto C está conformado por seis elementos y se tiene que: $\mathcal{A}(A,B,C) = V_C = [0,1,0,0,1,0]$ se deberá interpretar que el algoritmo detecta como posibles ARNs-lnc al segundo y quinto elemento del conjunto C. Para algunas combinaciones de conjuntos A, B y C es posible que las desigualdades $N_A \geq l_4$ y $N_B \leq l_5$ no se satisfagan. En este caso, se tiene que: $\mathcal{A}(A,B,C) = V_C = [0,0,0,0,0,0]$. Además, nótese también que este resultado puede ser obtenido aún cuando las desigualdades se satisfagan. Una representación esquematizada de este algoritmo puede ser consultada en la Figura 4.3.

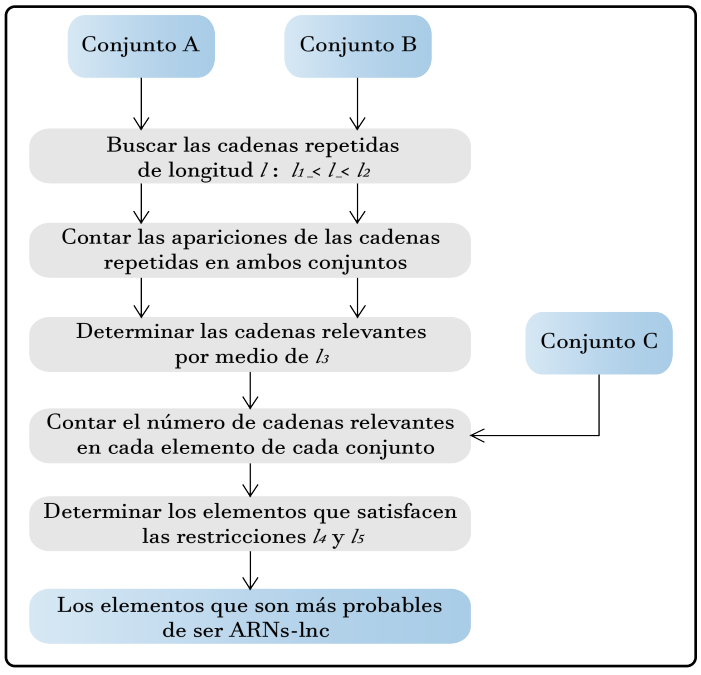


Figura 4.3: Una descripción gráfica del proceso computacional para buscar cadenas repetidas en gráficas de árbol con raíz asociadas a ARNs-lnc.

En general, dado tres conjuntos A, B y C, el algoritmo propuesto permite hacer una posible clasificación de los elementos en el conjunto C, basándose en el análisis de las similitudes encontradas en cada conjunto analizado, A y B, y del análisis de la relación que existe entre las similitudes de A y B. Buscando clarificar la manera en que se analizan los datos con el algoritmo propuesto, en la siguiente sección se expondrá un ejemplo hipotético para ello.

4.2. Ejemplo práctico del algoritmo

En esta sección se expondrá un ejemplo conciso para ilustrar cómo funciona el algoritmo propuesto en la sección 4.1. El algoritmo busca y compara las *NPPS* de cada grupo control buscando las cadenas más largas que se repiten. Posteriormente, determina cuáles de estas cadenas no están presentes en el otro conjunto, estableciendo con ello cuáles son las cadenas más relevantes. Con ello, se busca diferenciar un grupo de otro.

Para poder usar el algoritmo, se establecen dos grupos control, conformados por cuatro gráficas de árbol, mostradas en la Figura 4.4, y sus *NPPS* correspondientes, mostradas en el Cuadro 4.1, $S_1 = \{T1, T2, T3, T4\}$ y $S_2 = \{U1, U2, U3, U4\}$. Cabe señalar que si bien estas gráficas de árbol no provienen de una secuencia real de ARN servirán para ejemplificar la función del algoritmo.

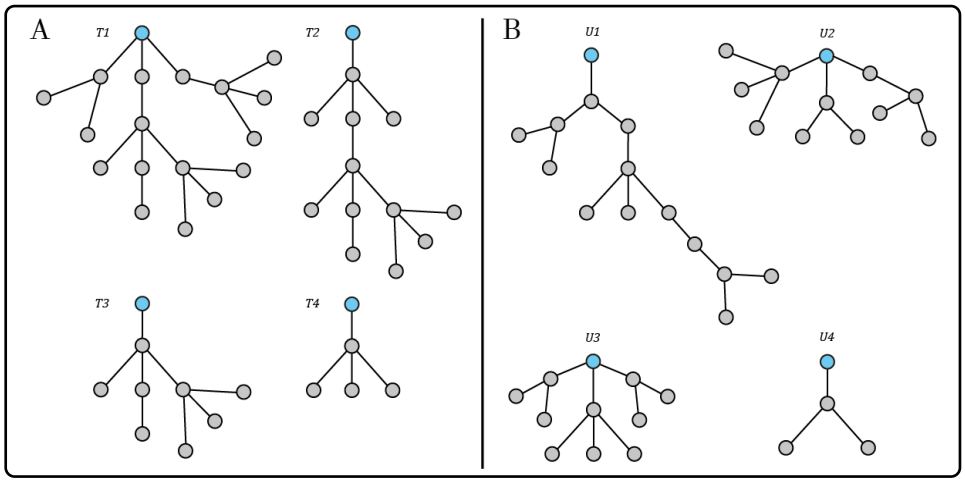


Figura 4.4: Grupo control S_1 (A) y Grupo control S_2 (B).

Cuadro 4.1: Las *NPPS* para analizar en los grupos control S_1 y S_2.

	Grupo control S_1
$T1$	.(.(.(.).(.).(.).).).(.(.(.(.).(.).(.).).(.(.).).(.).).).(.(.).(.).).
$T2$	.(.(.).(.(.(.).(.).(.).).(.(.).).(.).).).(.).).
$T3$	.(.(.(.).(.).(.).).(.(.).).(.).).
$T4$	.(.(.).(.).(.).).

	Grupo control S_2
$U1$	.(.(.(.(.(.(.(.).(.).).).).(.).(.).).).(.(.).(.).).
$U2$	.(.(.(.).(.).).(.(.).(.).(.(.).(.).(.).).
$U3$	.(.(.).(.).).(.(.).(.).(.).).(.(.).(.).).
$U4$	.(.(.).(.).).

Ahora, se establecen los parámetros: $l_1 = 6$ y $l_2 = 41$ para buscar cadenas de longitud mínima 6 y longitud máxima 41 y, con ello, el conjunto de cadenas repetidas en S_1, RS_1, y del mismo modo en S_2, RS_2, compuestos por 9 elementos cada uno. Los conjuntos con las cadenas repetidas encontradas se muestran en el Cuadro 4.2.

Cuadro 4.2: Los conjuntos con las cadenas repetidas en los grupo control S_1 y S_2.

	RS_1	RS_2
1	.).(.(.(.(.).(.).(.).).(.(.).).(.).).).(.	.(.(.).(.).).(.(.).(.).(.).).
2	.(.(.(.).(.).(.).).(.(.).).(.).).	.(.).(.).).).(.(.).(.).).
3	.(.(.(.).(.).(.).).	.).).(.(.).(.).).
4	.(.(.).(.).(.).).	.(.(.(.).(.).).).
5	.).(.).).).(.	.(.(.).(.).).
6	.(.(.).(.	.).(.).(.).).
7	.).).(.(.	.).).).(.
8	.(.(.(.	.(.(.(.
9	.).(.(.	.).).(.

Cabe señalar, que en el Cuadro 4.2 hay una cadena que está presente tanto en S_1 como en S_2. A saber: $RS_1(8)$ también está presente en RS_2. Así, dicha cadena no sería útil si se busca distinguir un elemento de S_1 de S_2.

Luego, se establecen las cadenas más relevantes en RS_1. Para ello, se crean dos matrices que cuantifican el número de apariciones de las cadenas RS_1 en S_1 y en S_2, $MRS_{1,1}$ y $MRS_{1,2}$, respectivamente.

$$MRS_{1,1} = \begin{pmatrix} 1 & 1 & \mathbf{2} & 2 & 2 & 3 & 3 & 2 & 3 \\ 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 2 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \quad MRS_{1,2} = \begin{pmatrix} 0 & 0 & \mathbf{0} & 0 & 1 & 2 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 & 3 & 2 & 1 & 2 \\ 0 & 0 & 0 & 1 & 0 & 3 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Por ejemplo, la entrada $3,1$ de la matriz $MRS_{1,1}$ es el número de veces que la cadena repetida tres de RS_1, **.(.(.(.).(.).(.).).).** , está presente en la primera cadena de S_1, a saber 2 veces en $T1$. Mientras que, la entrada $3,1$ de la matriz $MRS_{1,2}$, muestra que la cadena repetida tres de $RS1$, **.(.(.(.).(.).(.).).).** , no está presente en la primera cadena de S_2, a saber sin aparición en $U1$. De manera similar, se cuantifica el número de apariciones de las cadenas RS_2 en S_1 y en S_2, $MRS_{2,1}$ y $MRS_{2,2}$, respectivamente.

$$MRS_{2,1} = \begin{pmatrix} 0 & 1 & 1 & 1 & 2 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 2 \\ 1 & 0 & 1 & 0 & 2 & 1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad MRS_{2,2} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 2 & 2 & 2 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Finalmente, se comparan las matrices $MRS_{1,1}$ y $MRS_{1,2}$ ($MRS_{2,1}$ y $MRS_{2,2}$) para establecer qué cadenas en RS_1 (RS_2) son más relevantes en S_1 (S_2) y, consecuentemente, menos

importantes en S_2 (S_1), esto con base en el discriminante $l_3 = 2$. Las cadenas relevantes obtenidas son mostradas en el Cuadro 4.3. Cabe señalar que, si se fija el valor de $l_3 = 0$ a las cadenas mostradas en el Cuadro 4.3, se deberían agregar 5 cadenas más en el conjunto S_1: 5, 6, 7, 8 y 9, y 4 cadenas más en el conjunto S_2: 6, 7, 8 y 9. Es decir, dado que no se está restringiendo a que exista una diferencia entre el número de apariciones de dicha cadena en S_1 y la misma cadena en S_2, se deberían entonces tomar en cuenta todas las cadenas sin hacer distinción; así, se tendrían las cadenas del Cuadro 4.2. El elegir un valor de $l_3 > 0$ permite establecer las cadenas que son más importantes en un conjunto que en otro.

Cuadro 4.3: Cadenas relevantes compartidas con $l_3 = 2$ en S_1 y S_2.

	RS_1	RS_2
1	.).(.(.(.(.).(.).(.).).(.(.).(.).).(.).	.(.(.).(.).).(.(.).(.).(.).).
2	.(.(.(.).(.).(.).).(.(.).).(.).).	.(.).(.).).).(.(.).(.).).
3	.(.(.(.).(.).(.).).	.).).(.(.).(.).).
4	.(.(.).(.).(.).).	.(.(.(.).(.).).).
5		.(.(.).(.).).

Así, se ha establecido cuáles de las cadenas que comparten los conjuntos S_1 y S_2 son más relevantes. Con ello, las cadenas relevantes de S_1 (S_2) no lo son en S_2 (S_1). Por lo tanto, si se buscara establecer si una gráfica pertenece al conjunto S_1 o S_2 se deberían buscar las cadenas importantes en dicha gráfica y analizar el número de apariciones de ellas, para finalmente, concluir a qué grupo pertenecen. Por ejemplo: dada la gráfica $G1$, mostrada en la Figura 4.5 (A), se puede notar que algunas de las cadenas que la componen son muy similares a las que componen a las gráficas del conjunto S_2; es decir, comparten las cadenas importantes en S_2 del Cuadro 4.3, RS_2: 3 y 5. Además, que ninguna de las cadenas importantes del conjunto S_1 están presentes en $G1$. Así, se puede concluir que la gráfica $G1$ es más probable que pertenezca al conjunto S_2. Del mismo modo, dada la gráfica $G2$, mostrada en la Figura 4.5 (B), se puede notar que comparten las cadenas importantes en S_1 del Cuadro 4.3, RS_1: 3 y 4. Por lo tanto, se puede concluir que la gráfica $G2$ es más probable que pertenezca al conjunto S_1. Cabe señalar que ninguna de las cadenas importantes de S_2 están presentes en la gráfica $G2$. En conclusión, dicho algoritmo permite establecer una posible clasificación, basándose en las cadenas encontradas en cada conjunto y analizando la relación entre ellas, lo que permite hacer una clasificación con base en su estructura.

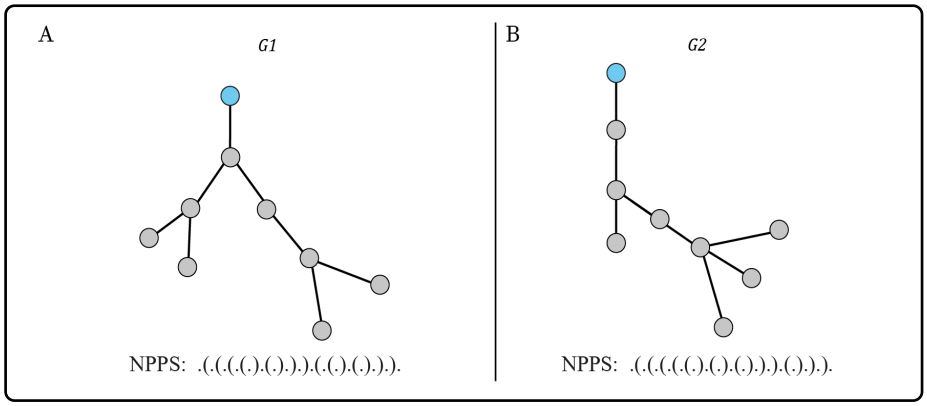


Figura 4.5: Gráfica $G1$ con su *NPPS* (en A) y gráfica $G2$ con su *NPPS* (en B).

Ahora, se aplicará esta metodología y algoritmo en el Capítulo 5 a gráficas provenientes de secuencias de ARNs para poder establecer, dados dos grupos control A y B, si algún elemento de un tercer conjunto, grupo de prueba C, pertenece a un grupo o al otro.

Capítulo 5

Identificando ARNs-lnc

Dado que el objetivo de este trabajo es realizar un análisis detallado de las gráficas de árbol con raíz asociadas a algunas cadenas de ARNs con el fin de extraer información estructural y, además, analizar si estas estructuras están presentes en otros grupos de ARNs, para con ello poder identificar ARNs-lnc. En el Capítulo 4, se propuso un algoritmo que busca hacer una clasificación entre diferentes grupos de ARNs mediante el análisis de sus correspondientes estructuras secundarias. Este analiza grupos de secuencias de ARNs por medio de sus gráficas de árbol con raíz en *NPPS* y algunas medidas estadísticas, con la finalidad de extraer su información estructural. El objetivo de dicho análisis es identificar, comparar y clasificar las estructuras presentes en los grupos analizados. De forma general, este algoritmo busca establecer las cadenas que aparecen con más frecuencia en dichos grupos y las compara. Para con ello, caracterizar un elemento de un grupo en particular. La Figura 5.1 muestra de manera simplificada el proceso que se lleva a cabo para determinar un ARN-lnc. Dicho algoritmo se implementó usando Python v.3.8, podrá encontrar el código del algoritmo y un manual para su uso en la Capítulo 6. Además, podrá encontrar el pseudocódigo de dicho algoritmo en el Apéndice E.

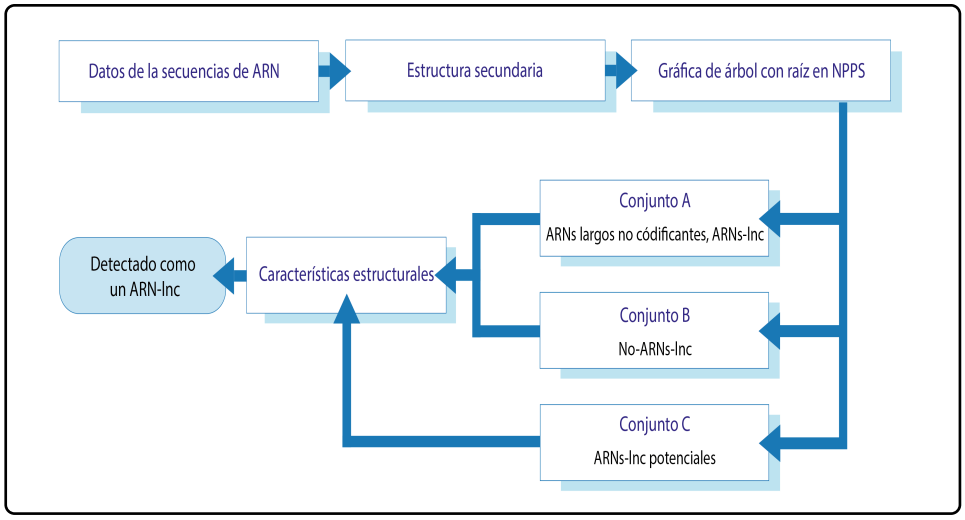


Figura 5.1: Este diagrama muestra de manera simplificada el proceso para determinar un ARN-lnc.

5.1. Caso: ARNs-lnc de *S. cerevisiae*

En la sección 2.1.1 se estableció la importancia de estudiar los ARNs-lnc, mientras que en la sección 2.1.3 se abordó la importancia de la levadura *S. cerevisiae*. Con este fin, utilizando el algoritmo propuesto en la sección 4.1 se hará el análisis correspondiente. Así, siguiendo este camino, para ser analizados se establecieron tres conjuntos de ARNs, sección 2.1.4, todos pertenecientes a dicho organismo. A manera de recordatorio:

• **Grupo control** $\bar{A}$: constó de las únicas 18 secuencias registradas en la literatura [46], hasta el año 2019, como ARNs-lnc.

• **Grupo control** $\bar{B}$: constó de 18 secuencias seleccionadas aleatoriamente de un conjunto de 64 ARNs que no pertenecen al conjunto de ARNs-lnc, ARNs-otros.

• **Grupo de prueba** $\bar{C}$: constó de 6 secuencias de ARNs desconocidos, los cuales fueron proporcionados por la Dra. Lina Raquel Riego Ruiz y su equipo de la División de Biología Molecular, IPICYT, de las cuales se sabía que tres de ellas no pertenecían al conjunto de ARNs-lnc, mientras que las tres restantes eran potencialmente ARNs-lnc.

A pesar de que miles de ARNs han sido computacionalmente identificados como posibles candidatos para ser ARNs-lnc en *S. cerevisiae*, solo a un pequeño subconjunto de ellos se les ha podio confirmar, por medio de evidencia experimental, su función. Para este análisis se seleccionaron 18 ARNs-lnc validados de NONCODE (http://www.noncode.org/keyword.php), SGD [46] y de la literatura (Balarezo-Cisneros *et al.* [48]; Xu *et al.* [61]; van Dijk *et al.* [62]; Geisler *et al.* [63]; Castelnuovo *et al.* [64]. En conclusión, esta elección prioriza una buena caracterización de ARNs-lnc sobre el tamaño del conjunto de datos, minimizando así la inclusión de transcripciones inciertas.

Por otro lado, como se mencionó anteriormente, utilizando NUPACK Web con los conjuntos $\bar{A}$, $\bar{B}$ y $\bar{C}$, se obtienen las estructuras secundarias en *NPP* de cada conjunto, $\hat{A}$, $\hat{B}$ y $\hat{C}$, respectivamente. Luego, se utilizan las reglas de Gan para obtener los conjuntos en *NPP*, A_G, B_G, y C_G, respectivamente, buscando establecer sus gráficas de raíz asociadas. Finalmente, empleando el método establecido en la sección 3.3, se simplifica su *NPP* a su *NPPS*, conjuntos A, B, y C, respectivamente. El Cuadro 5.1 muestra los nombres de los ARNs que conformaron cada grupo, la parte inicial de su secuencia en *NPPS*, así como la longitud original de la secuencia ($|\bar{L}|$) y la longitud de su *NPPS* ($|L|$).

Cuadro 5.1: Datos de entrada para el algoritmo, conjuntos: A, B y C.

Grupo control $\bar{A}$			
ARN	NPPS	$\lvert\bar{L}\rvert$	$\lvert L\rvert$
ICR1	.(.).(.(.).).(.).(.(.(.(.(.(.(.).(.(.(.(.).(.(.(.(.).(.(.(.(.).).).).(.(.(.(.(.).).).).).(.(.(.).(.).(.(.).(.).).).).) ···	3199	625
RME2	.(.).(.(.).).(.).(.(.(.(.(.(.(.(.(.).(.(.(.(.(.(.).).).).).).(.(.).).(.).(.(.(.(.).).).).(.).(.(.).).(.) ···	2223	461
RME3	.(.).(.).(.(.(.(.).(.(.).).).(.(.(.).).).).(.(.).(.(.(.(.).).(.(.(.(.).).).).).).(.(.(.(.(.(.(.(.(.(.).) ···	1905	373
IRT1	.(.).(.(.).).(.(.(.(.).).).).(.).(.(.).).(.(.(.(.(.(.(.(.).).).).).(.).(.).(.(.(.).).(.(.(.(.(.(.(.(.(.(.(.(···	1489	309
TLC1	.(.(.(.(.(.(.(.(.(.(.(.(.(.(.(.(.(.(.(.).(.).(.(.(.).).(.(.(.).).).).).).).).).).).).).).(.(.(.).).).(.(.(.(···	1301	265

PWR1	.(.).(.(.(.).(.(.(.(.).).).).).(.).(.(.(.(.(.(.).).(.).).).).).(.).(.).(.(.(.(.(.).).(.(.(.).).).).(.(. ⋯	941	181
RUF5-1	.(.).(.).(.).(.).(.(.).(.(.(.(.(.).(.(.(.(.).).).).).(.).(.(.(.(.(.(.).).).).(.(.).).).).(.).).).).(.(.).(.(.). ⋯	710	153
RUF21	.(.(.).).(.(.(.(.).).).).(.(.(.(.(.(.).(.(.(.).(.(.(.(.(.).(.(.).(.).(.).).).).).(.).).(.).(.(.).(.(. ⋯	707	149
ETS1-1	.(.(.(.).).(.).(.(.).).(.).(.(.(.).).).(.(.).(.(.(.).).(.(.(.(.).).).).(.(.(.(.).).).).(.).(.(.(.(.).).(.(.). ⋯	700	133
SRG1	.(.(.(.).).(.).(.(.(.(.(.(.).(.(.(.).).).(.(.).(.).).).).).).(.(.(.(.(.).).).).).(.).(.).(.).(.).	551	109
RUF22	.(.).(.).(.).(.).(.).(.(.(.).).).(.(.(.(.).).).).(.).(.(.).(.(.(.(.).).).).(.(.(.).).).	515	93
RUF20	.(.(.(.(.(.).(.(.(.(.(.(.).(.).).).).).(.(.).).).).).(.(.).(.).).(.).(.).	443	89
ITS1-1	.(.(.(.(.(.(.).).).).).).(.).(.(.).).(.(.(.).).).	361	53
RUF23	.(.).(.(.(.(.(.).).).).).(.).(.(.).).(.(.(.).).).(.).	254	53
ITS2-1	.(.(.).).(.(.(.(.).).).).(.).(.(.).).(.(.).).(.).	232	49
ETS2-1	.(.).(.).(.(.(.(.(.).).).).).(.(.(.).).).	211	45
RNA170	.(.(.(.(.).).).).(.).).(.).(.).	169	33
ZOD1	.(.).(.).	58	9

Grupo control $\bar{B}$			
ARN	NPPS	$\lvert\bar{L}\rvert$	$\lvert L\rvert$
15S Ribosomal	.(.).(.(.(.(.(.).).).).(.).(.).(.).(.).(.(.).).(.).(.).(.).(.).(.(.(.).).(.).(.).(.(.).(.).(.).).(.).(.(.(.(⋯	1649	317
LSR1	.(.(.(.).).(.(.).(.(.).).).(.).(.). ⋯	1175	185
Telomerase	.(.(.(.(.(.(.(.(.(.(.(.(.(.(.(.).(.).(.(.(.).).(.(.(.).).).).).).).).).).).).).(.(.(.(.).).).(.(.(.(.). ⋯	1158	241
SNR86	.(.).(.(.(.(.).).).).).).).).).).).).).).).).).). ⋯	1004	157
SNR30	.(.(.(.).).(.(.).(.(.(.).(.(.(.(.(.(.(.(.).).).).).).).).(.).).).(.).(.(.(.(.(.).).).).(.(.).(.(.).).	609	113
Small nuclear SNR30	.(.(.(.).).(.(.).(.(.(.).(.(.(.(.(.(.(.(.).).).).).).).).(.).).).(.).(.(.(.(.(.).).).).(.(.).(.(.).).	606	113
SNR19	.(.).(.).(.(.(.(.).).).(.(.).).(.(.(.(.(.(.).).).).).(.(.(.(.(.(.(.(.).).).).).(.(.).(.).).).(.).	568	113
SNR84	.(.(.(.).(.(.).).).(.(.).(.(.).(.).(.).(.(.(.(.(.(.(.).(.).(.(.(.(.).).).).).).).).).).).(.).(.).	550	109
Small nuclear SNR84	.(.(.(.).(.(.).).).(.(.).(.(.).(.).(.).(.(.(.(.(.(.(.).(.).(.(.(.(.).).).).).).).).).).).(.).	537	105
RPM1	.(.).(.(.).(.(.(.).).).(.).(.).(.).(.(.(.).).).(.).(.).(.(.).).	483	73
SNR17B	.(.).(.(.(.).).(.).(.).(.).(.(.).).(.(.(.(.(.(.(.).).).).(.(.).).(.).(.(.).).).).	462	93
Nuclear RNASE P	.(.(.(.(.).).).(.(.(.(.).(.).(.(.).(.(.).(.).(.(.(.).).).).).).	358	77
SNR42	.(.).(.(.(.(.).).).(.(.(.).).(.).(.).(.(.(.(.).).).).	351	57
NME1	.(.(.(.).(.(.(.).(.(.).).(.).(.(.(.(.).).).).).).	340	57
U3	.(.(.(.(.(.).).).).).(.(.(.).).).(.(.).(.(.(.(.(.).).).).	334	69
Small nuclear U3	.(.).(.(.).(.(.(.(.(.(.(.).).).).(.(.).(.).(.(.).).).).	333	69
RNASE MRP	.(.(.(.).(.(.(.).(.(.).).(.).(.(.(.(.).).).).).).	332	57
SNR83	.(.).(.(.).(.).(.(.).).(.(.).(.).	306	33

Grupo de prueba $\bar{C}$			
ARN	NPPS	$\lvert\bar{L}\rvert$	$\lvert L\rvert$
KAP123	.(.(.(.).(.(.(.(.).).).(.(.(.(.(.(.).(.).).(.(.(.(.(.(.(.).).).(.(.(.(.(.(.).).).).(.(.).).(.(.(. ⋯	3342	725
GRE2	.(.(.(.).(.(.).(.).(.(.(.).(.(.(.).).).).).).(.(.(.(.(.).(.(.(.(.).).).(.(.(.).).).(.(.(.).). ⋯	1029	229
ECM11	.(.).(.).(.).(.(.).).(.).(.(.(.(.).).).(.(.(.).).(.(.(.).).(.(.(.).).).(.(.).(.).(.(.(.(.(.(.(.(.(.(. ⋯	909	173
1477	.(.(.(.(.(.).(.).(.(.(.(.(.(.(.).).(.).(.(.(.).).(.(.(.(.(.(.).).(.).(.(.).).).).).).(.).).(.(.(.(.(. ⋯	843	169
6754	.(.).(.).(.(.).).(.(.).).(.(.(.(.(.).(.(.(.(.).).).).).).).).(.(.(.(.(.(.).).(.(.(.).).).).).).).).(.(⋯	840	161
12189	.(.(.(.(.(.).(.).(.(.(.(.(.).).(.(.).(.).).).).).).(.(.(.(.).(.(.(.).).).).).).).).).(.).	583	101

Cabe señalar que existe una diferencia entre la longitud original en *NPP* asociada a una estructura secundaria de ARN y la cadena final que se obtiene con el método establecido en la sección 3.3, *NPPS*. La *NPPS* es de menor longitud en comparación con la *NPP*; sin embargo, preserva la información importante de la estructura secundaria y además representa una gráfica de árbol con raíz.

Los resultados utilizando estos datos de entrada para el algoritmo y las correspondientes discusiones se podrán encontrar en las secciones 5.1.1 y 5.1.2.

5.1.1. Analizando los conjuntos A, B y C

Para llevar a cabo dicho análisis se usaron los tres conjuntos A, B y C descritos en el Cuadro 5.1. Los conjuntos A y B se tomaron como los grupos control, de los cuales nos interesa saber si comparten alguna característica en particular. Mientras que, el conjunto C se tomó como el grupo de prueba. En este caso, los parámetros usados en el algoritmo se calcularon de manera heurística de tal forma que, el algoritmo detectará al menos 7 elementos del conjunto A y a lo más 3 elementos del conjunto B. En este caso, los parámetros utilizados fueron $l_1 = 22$, $l_2 = 34$, $l_3 = 2$, $l_4 = 7$, $l_5 = 3$. Donde: l_1 es la longitud mínima y l_2 la longitud máxima de la cadena a buscar, l_3 es el discriminante para tratar de caracterizar cada conjunto, l_4 el número mínimo en A que se busca detectar y l_5 el número máximo en B que se permiten detectar. Con estos datos de entrada se obtuvo:

$$\mathcal{A}(A,B,C,l_1,l_2,l_3,l_4,l_5) = \mathcal{A}(A,B,C,22,34,2,7,3) = V_C = [0,1,0,1,1,1]$$

Recordar que, dados los conjuntos A, B, y C, el vector resultante mediante el algoritmo $\mathcal{A}(A,B,C)$ es: $V_D = V_A + V_B + V_C$. Pero, dado que los resultados en V_A y V_B son acotados por los parámetros l_4 y l_5, nos permiten enfocarnos en los candidatos a ser probables ARNs-lnc, información contenida en V_C. Cabe señalar que a mayor cantidad de elementos detectados en V_A, y a su vez la menor cantidad de elementos detectados en V_B, la conclusión en V_C será más precisa. Así, es importante enfocarse en el vector V_C que conforma al vector resultante V_D.

Analizando el vector resultante V_C, se tiene que los ARNs en el conjunto C más probables de ser ARNs-lnc son *GRE2*, *1477*, *6754* y *12189*. Se sabía con antelación que el ARN *GRE2* no pertenecía al conjunto de ARNs-lnc, obteniendo con ello un falso positivo. Por otro lado, el algoritmo sugiere que los ARNs *1477*, *6754* y *12189* son probablemente ARNs-lnc.

Cabe señalar que para establecer el vector resultante V_C, el algoritmo identifica 13 elementos del conjunto A y tres elementos del conjunto B. A saber:

En el conjunto A fueron detectados los ARNs *ICR1*, *RME3*, *PWR1*, *RUF5-1*, *RUF21*, *ETS1-1*, *SRG1*, *RUF22*, *RUF20*, *ITS1-1*, *RUF23*, *ITS2-1*, *ETS2-1*. Lo cual es una buena detección dado que solo los ARNs *RME2*, *IRT1*, *TLC1*, *RNA170* y *ZOD1* no fueron detectados, de los cuales los últimos dos ARNs *RNA170* y *ZOD1* quedan fuera de la búsqueda, ya que se buscan cadenas de longitud entre 22 y 34 caracteres, dichos ARNs tienen 33 y 9 caracteres de longitud, respectivamente. Así, se tiene el vector:

$$V_A = [1,0,1,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0].$$

Mientras que en el conjunto B fueron detectados solo 3 ARNs *LSR1*, *Nuclear RNASE P* y *U3*; es decir, solo tres falsos positivos. Así, se tiene el vector:

$$V_B = [0,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0].$$

En resumen, el 72.2 % de los ARNs en el conjunto A, fueron detectados correctamente como ARNs-lnc. Mientras que en el conjunto B el 83.3 % de los ARNs fueron detectados correctamente. Cabe señalar que si un ARN en el conjunto B no es identificado como un ARN-lnc nos indica también una correcta identificación. Finalmente, con esta información podemos concluir que los ARNs establecidos en el vector V_C son probablemente ARNs-lnc.

Finalmente, vale la pena notar que Ana Novaičić y colaboradores en [47] validaron conforme a un experimento que el ARN *12189* es un ARN-lnc, lo cual da certeza a la metodología establecida y valida los resultados. Sin embargo, buscando establecer una mejor detección de ARNs-lnc se buscará establecer un procedimiento que permita compensar los pocos datos que se tienen para el análisis, para con ello robustecer el algoritmo y, por ende, la predicción.

5.1.2. Analizando subconjuntos de los conjuntos A, B y C

Como en el caso anterior, sección 5.1.1, para llevar a cabo dicho análisis se usaron los tres conjuntos A, B y C descritos en el Cuadro 5.1. Los conjuntos A y B se tomaron como los grupos control, de los cuales nos interesa saber si comparten alguna característica en particular. Mientras que, el conjunto C se tomó como el grupo de prueba. Dado que se tienen pocos datos disponibles, para validar el análisis, se seleccionaron de manera aleatoria 30 subconjuntos con 12 elementos del conjunto A, A_i, y 30 subconjuntos de solo 3 elementos de A que fueron parte del conjunto de prueba C, CA_i, donde $i = 1,2,\ldots,30$. Cabe señalar que los conjuntos A_i y CA_i son disjuntos. De manera análoga, se obtienen los 30 subconjuntos del conjunto B, B_j y CB_j, donde $j = 1,2,\ldots,30$. Cabe remarcar que el conjunto C estará formado por un CA_i y un CB_j. En este caso, los parámetros usados en el algoritmo se calcularon de manera heurística de tal forma que el algoritmo detectará al menos 7 elementos del conjunto Ai y a lo más 3 elementos del conjunto Bi. En este caso los parámetros utilizados fueron $l_1 = 22$, $l_2 = 34$, $l_3 = 2$, $l_4 = 7$, $l_5 = 3$. Donde: l_1 es la longitud mínima y l_2 la longitud máxima de la cadena a buscar, l_3 es el discriminante para tratar de caracterizar cada conjunto, l_4 el número mínimo en A que se busca detectar y l_5 el número máximo en B que se permiten detectar. Cabe señalar que el discriminante $L_3 = 2$ se eligió después de observar que, en este caso, los valores de los elementos en los vectores $C_{AA} - C_{AB}$ y $C_{BB} - C_{BA}$ suelen oscilar en el rango de cero a seis. A medida que aumentamos el valor de dicho umbral, el número de cadenas asociadas disminuye. Mientras que para valores de $l_3 \leq 1$ las cadenas encontradas estarían la misma cantidad de veces en A y en B, lo que claramente no permitiría hacer una clasificación entre estos dos conjuntos.

Ahora, sea K el vector que realizará el seguimiento de los ARNs-lnc detectados. Así, para el cálculo del vector K se iteran los i-ésimos y j-ésimos elementos de los conjuntos A_i, B_j, $CA_i + CB_j$:

$K = [0,0,0,0,0,0]$
para i en el rango (30):
    para j en el rango (30):
        $K = K + \mathcal{A}(A_i, B_j, CA_i + CB_j)$

Claramente, los elementos en CA_i deben ser más probables de ser detectados como ARNs-lnc, mientras que los elementos en CB_j no lo debieran ser. El algoritmo $\mathcal{A}$ se puso a prueba 900 veces y se sumaron los vectores obtenidos para obtener el vector resultante $V_C = [170,381,259,41,2,195]$, lo cual significa que el segundo elemento es detectado como un ARN-lnc 381 veces, mientras que el quinto elemento solo se detectó dos veces. Recordar

que en el conjunto C sus elementos varían en relación con i y j, así $CA_i + CB_j$ es diferente en cada iteración. Pero, en el resultado general de detectar los elementos más probables de ser ARNs-lnc, se tiene que el 77.3 % de las veces la predicción fue correcta, en contraste con el 22.7 % en que no lo fue.

Por otro lado, si por ejemplo se fija un valor para el conjunto C con $i = 3$ y $j = 5$, es decir, si se calcula $\mathcal{A}(A_i, B_j, CA_3 + CB_5)$ variando i y j en A y B, se obtiene el vector resultante $V_C = [370, 406, 597, 149, 0, 0]$, el cual indica que el tercer elemento de CA_3, tercer elemento de V_C, fue correctamente detectado como un ARN-lnc 597 veces de las 900 veces que se puso a prueba el algoritmo, mientras que el segundo elemento de CB_5, quinto elemento de V_C, no fue detectado como un ARN-lnc ninguna vez. Lo cual implica que el 90.21 % de las veces los elementos en CA_3 fueron correctamente detectados como ARNs-lnc. Del mismo modo, poniendo a prueba el algoritmo con otros conjuntos fijos $CA_{i_0} + CB_{j_0}$ se obtuvo que el mayor número de las veces un elemento del conjunto CA_{i0}; es decir, de las primeras tres posiciones de V_C puntuaba con el mayor número de veces detectado como un ARN-lnc.

Ahora bien, si se analiza solo el elemento con el mayor número de veces asignado como ARN-lnc y este se asocia a un nuevo vector, V_R, en el cual se asigne el número 1 para indicar que el elemento fue detectado como un ARN-lnc el mayor número de veces. Por ejemplo, en el caso $V_C = [70, 406, 597, 149, 0, 0]$, se tomará como el ARN-lnc con el mayor número de veces detectado el elemento tres de V_C obteniendo así el vector $V_R = [0, 0, 1, 0, 0, 0]$. Así, por ejemplo, si ahora se pone a prueba el algoritmo 500 veces y se varía el conjunto $CA_{3j} + CB_{5j}$, para $j = 1, \ldots, 500$, y por cada iteración se suma el vector obtenido V_R se tendrá como resultado el vector $V_R = [149, 170, 181, 0, 0, 0]$, el cual indica que las 500 veces las secuencias detectadas están dentro del conjunto de los ARNs-lnc, y al mismo tiempo, que ningún elemento del conjunto de los ARNs que no pertenecen a los ARNs-lnc fueron detectados como ARNs-lnc. Los resultados se muestran en la Figura 5.2 (A), donde A3_1, A3_2 y A3_3 representan elementos del conjunto A que cambian en cada iteración, pero se analiza su aparición acumulativa, en este caso, las 500 veces fueron detectados como probables ARNs-lnc, lo cual es correcto dado que son elementos del conjunto A.

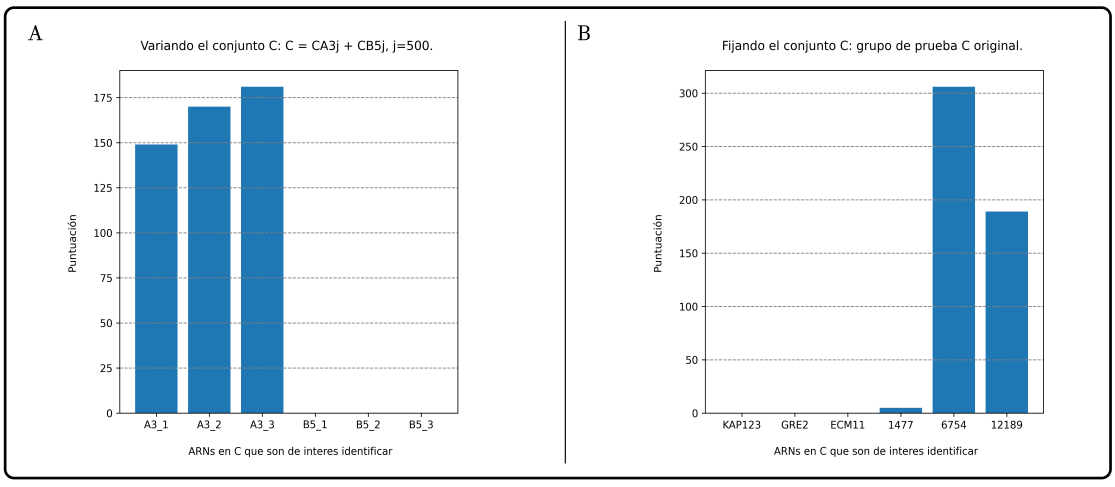


Figura 5.2: Los resultados del algoritmo después de 500 iteraciones: con $C = CA_{3j} + CB_{5j}$ (en A) y con los datos del conjunto C original (en B), usando NUPACK Web.

Consiguientemente, se fijó el conjunto C con los datos del conjunto C original y de manera similar se puso a prueba el algoritmo 500 veces, con el fin de establecer el número de veces que cada elemento del conjunto C es detectado como un ARN-lnc. Se obtuvo como resultado:

$$V_R = [0,0,0,5,306,189].$$

Así, los resultados indican que los ARNs *1477*, *6754* y *12189*, fueron detectados como ARNs-lnc el 1 %, 61.2 % y 37.8 % de las veces, respectivamente. Nótese que los otros tres ARNs no fueron detectados en ninguna ocasión como ARNs-lnc, lo cual es un buen resultado, dado que sabíamos de antemano que tres elementos de este conjunto no eran ARNs-lnc. Los resultados se muestran en la Figura 5.2 (B). Así, el algoritmo pudo detectar qué elementos no pertenecían al grupo de ARNs-lnc y estableció cuáles tienen mayor probabilidad de serlo; a saber, los ARNs *6754* y *12189*. Cabe señalar que dado a que en todas las pruebas del algoritmo los conjuntos A_i y B_j son tomados de manera aleatoria, los resultados varían tras cada prueba; sin embargo, la clasificación se preserva en la mayoría de los casos. Por ejemplo, poniendo a prueba una vez más el algoritmo, con los mismos datos del conjunto C original se obtiene como resultado:

$$V_R = [0,0,0,3,291,206].$$

Así, los ARNs *6754* y *12189* se mantienen como los más probables de ser ARN-lnc, mientras que los tres primeros elementos no lo son. Como se mencionó anteriormente, Ana Novaičić *et al.* en [47] validaron conforme a un experimento que el ARN *12189* es un ARN-lnc, lo cual da certeza a la metodología establecida y valida los resultados. Con ello, se sugiere priorizar el análisis del ARN *6754*, mediante ensayos en laboratorio, con el fin de determinar si es un ARN-lnc, dado que el algoritmo determina que dicho ARN tiene el mejor puntaje.

Finalmente, se determinó la complejidad temporal del algoritmo[1], teniendo en cuenta el tiempo promedio que tarda el algoritmo en realizar 100 pruebas cuando el número de subconjuntos tomados en cuenta varía entre $16, 18, 20, \ldots, 32$, podrá encontrar los resultados en la Figura 5.3. Donde, en color rojo se muestra la gráfica del polinomio $1.35x^2 - 12.90x + 172.38$ que mejor aproxima los puntos (x_i, y_i).

$$x = [16,18,20,22,24,26,28,30,32] \quad , \quad y = [308,367,448,579,639,743,850,972,1165].$$

Por lo tanto, la complejidad temporal del algoritmo para este conjunto de datos en particular es $O(n^2)$, *i.e.*, el tiempo de ejecución del algoritmo crece de manera cuadrática respecto al tamaño de la entrada n. Incluso, si aumentamos el número de ARNs en los conjuntos A, B y C, y mantenemos fijo el número de conjuntos utilizados para el bootstrapping[2] (en A y B: 12, y en C: 6) se esperan resultados similares a los mostrados en la Figura 5.3.

[1] La complejidad temporal establece una forma estandarizada de medir la eficiencia de un algoritmo a medida que aumenta el tamaño de sus datos de entrada.

[2] El bootstrapping es un método de remuestreo que permite estimar intervalos de confianza, donde las muestras sucesivas se extraen de nuestra muestra y no de la población de la que procede.

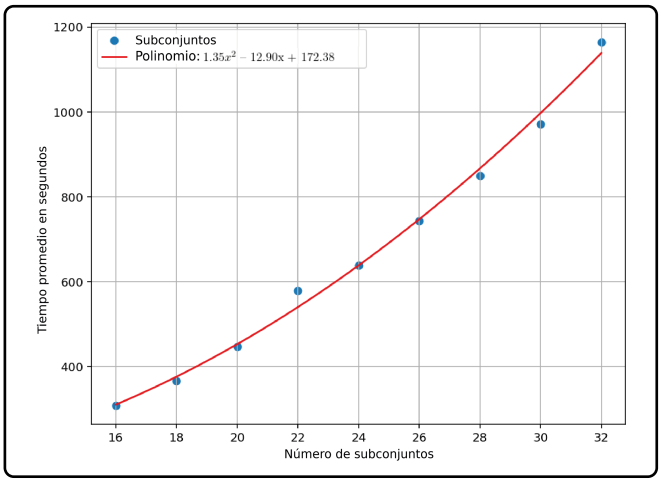


Figura 5.3: Gráfica del polinomio $1.35x^2 - 12.90x + 172.38$, que mejor aproxima los puntos (x_i, y_i).

Como se ha podido observar, el algoritmo nos permite hacer una clasificación usando como datos las *NPPS* asociadas a cada elemento de los conjuntos A, B y C. Recordar que antes de obtener las cadenas en *NPPS* se deben obtener las *NPP* asociadas a las estructuras secundarias de los ARNs para analizar. Dado que existen diversos programas para generar dichas estructuras, una pregunta que surge de manera natural es ¿qué pasa si en el algoritmo se cambia el programa para obtener las *NPP*?, ¿se podrá concluir algo? Pues bien, en la siguiente sección se explorará un poco sobre esta idea.

5.1.3. Empleando otros programas de plegamiento

Dado que diferentes programas de plegamiento pueden asociarle a un ARN una diferente estructura secundaria, se decidió usar el algoritmo siguiendo las ideas de la sección 5.1.2, pero ahora usando como programas de plegamiento RNAfold WebServer y rna-state-inf. Cabe resaltar que se usaron los mismos parámetros como en el caso donde se utilizó NUPACK Web:

$$\mathcal{A}(A_i, B_j, CA_i + CB_j, 22, 34, 2, 7, 3).$$

En la Figura 5.4 (A) se muestran los resultados usando RNAfold WebServer y el conjunto C, cómo $C = CA_{3j} + CB_{5j}$, es decir, ARNs-lnc y ARN que no pertenecen al grupo de ser ARNs-lnc, ambos conocidos. En este caso, se obtuvo como resultado:

$$V_R = [164, 183, 142, 3, 6, 2].$$

Donde A3_1, A3_2 y A3_3 representan elementos del conjunto A que cambian en cada iteración, pero se analiza su aparición acumulativa, en este caso, 489 veces fueron detectados como probables ARNs-lnc, lo cual es correcto dado que son elementos del conjunto A. Por otro lado, B5_1, B5_2 y B5_3 representan elementos del conjunto B que cambian en cada iteración, pero se analiza su aparición acumulativa, en este caso, 11 veces fueron detectados como probables ARNs-lnc, lo cual no es correcto, teniendo con ello 11 falsos positivos. Es

decir, el 97.8 % de las veces, los ARNs en el conjunto de ARNs-lnc fueron detectados correctamente, en contraste con las 11 veces, correspondientes al 2.2 % de las veces, los ARNs del conjunto que no pertenecían al conjunto de ARNs-lnc fueron erróneamente detectados.

En contraste, la Figura 5.4 (B) muestra los resultados usando RNAfold WebServer y el conjunto C original. En este caso, se obtuvo como resultado:

$$V_R = [9,0,0,0,0,491].$$

Lo cual sugiere que usando RNAfold WebServer, el ARN *12189* es el más probable de ser un ARN-lnc, en este caso.

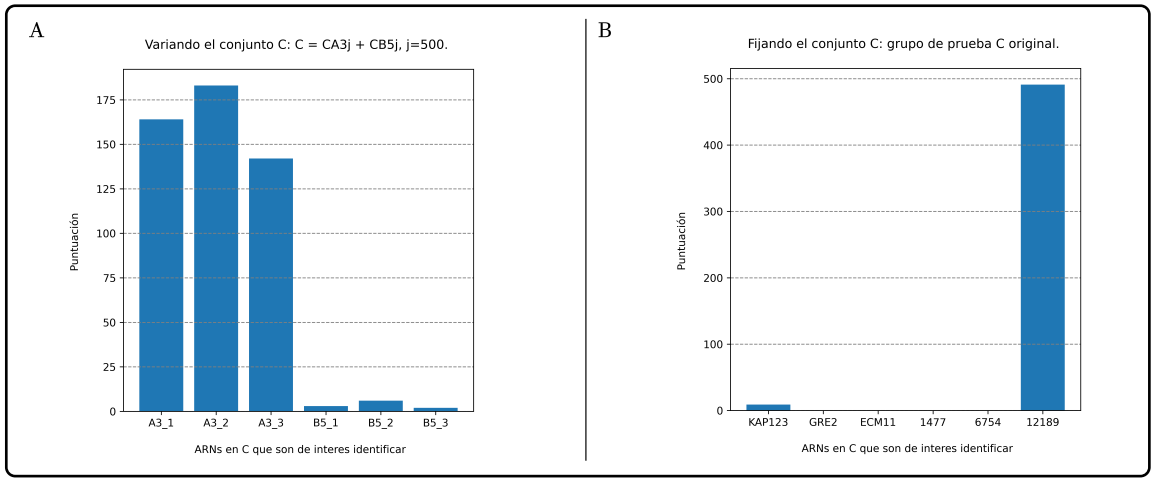


Figura 5.4: Los resultados del algoritmo después de 500 iteraciones: con $C = CA_{3j} + CB_{5j}$ (an A) y con los datos del conjunto C original (en B), usando RNAfold WebServer.

Por otro lado, en la Figura 5.5 (A) se muestran los resultados usando rna-state-inf y el conjunto C, como $C = CA_{3j} + CB_{5j}$; es decir, ARNs-lnc y ARN que no pertenecen al grupo de ser ARNs-lnc, ambos conocidos. En este caso, se obtuvo como resultado:

$$V_R = [183,158,155,2,1,1].$$

Donde A3_1, A3_2 y A3_3 representan elementos del conjunto A que cambian en cada iteración, pero se analiza su aparición acumulativa, en este caso, 496 veces fueron detectados como probables ARNs-lnc, lo cual es correcto dado que son elementos del conjunto A. Por otro lado, B5_1, B5_2 y B5_3 representan elementos del conjunto B que cambian en cada iteración, pero se analiza su aparición acumulativa, en este caso, 4 veces fueron detectados como probables ARNs-lnc, lo cual no es correcto, teniendo con ello 4 falsos positivos. Es decir, el 99.2 % de las veces, los ARNs en el conjunto de ARNs-lnc fueron detectados correctamente, en contraste con el 0.8 % que se detectaron falsamente como ARNs-lnc en el otro conjunto.

En contraste, la Figura 5.5 (B) muestra los resultados usando rna-state-inf y el conjunto C original. En este caso, se obtuvo como resultado:

$$V_R = [0,114,103,0,0,283].$$

Lo cual sugiere que el ARN *12189* es probable que sea un ARN-lnc. Cabe notar que los ARNs *GRE2* y *ECM11*, pertenecientes al conjunto de ARNs que no son ARNs del conjunto de ARNs-lnc, fueron erróneamente detectados, a saber 114 y 103 veces, respectivamente,

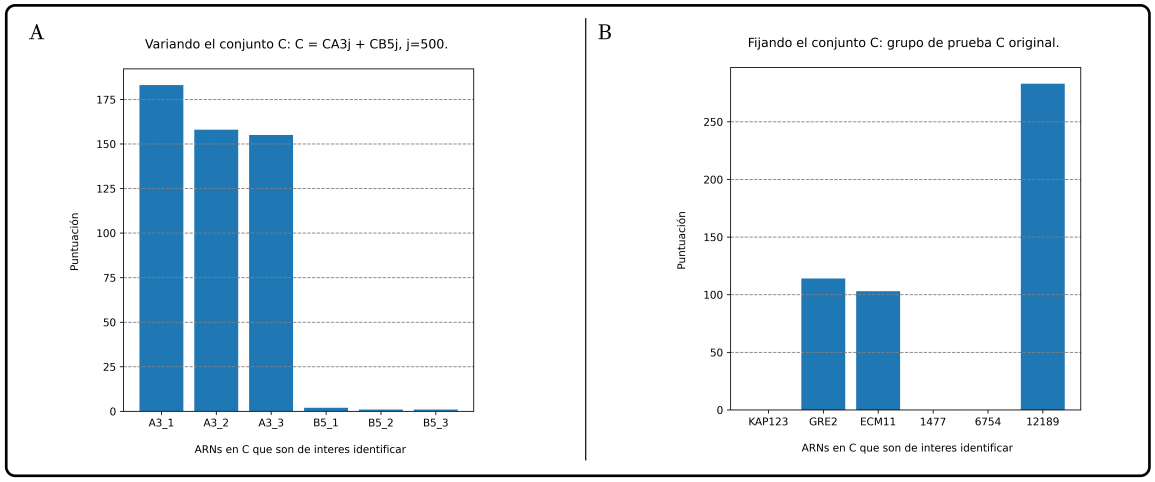


Figura 5.5: Los resultados del algoritmo después de 500 iteraciones: con $C = CA_{3j} + CB_{5j}$ (en A) y con los datos del conjunto C original (en B), usando rna-state-inf.

Si bien es cierto que existen más programas de plegamiento para generar las estructuras secundarias de un ARN, por ejemplo: mFold WebServer [53], mXfold2 [55], SPOT-RNA2 [65] o UFold [66], presentan algún inconveniente al emplearlos, para este trabajo no se pudieron obtener todas las secuencias del Cuadro 2.1 debido a sus longitudes. Para estos programas, la longitud de las secuencias a emplear está limitada a secuencias con 2400, 2000, 1000 y 600 nucleótidos, respectivamente. Es importante notar que en esta tesis no se busca establecer qué programa de plegamiento es mejor, sino más bien, mostrar que la detección de ARNs-lnc usando la metodología propuesta, sección 3.3, y el algoritmo, sección 4.1 se preservan aún cuando se usan diferentes programas para plegar las secuencias de ARN.

En todos los casos, los ARNs-lnc conocidos fueron identificados el 100 %, 97.8 % y 99.2 % de las veces, usando NUPACK Web, RNAfold WebServer y rna-state-inf, respectivamente, lo cual refuerza la robustez y confiabilidad del enfoque propuesto en esta tesis. Como se describió previamente, los parámetros l_i, con $i = 1,2,\ldots,5$, fueron establecidos heurísticamente usando NUPACK Web, los cuales se mantuvieron sin cambios al emplear el mismo procedimiento a RNAfold WebServer y rna-state-inf. Sin embargo, se espera que un mejor ajuste de dichos parámetros, de manera particular, para cada programa de plegamiento podría mejorar su desempeño de clasificación. Por otro lado, se puede observar una variación en los resultados referentes al grupo de prueba C usando los tres programas de plegamiento, la cual es consistente con el hecho de que diferentes algoritmos de plegamiento generan distintas estructuras secundarias y, consecuentemente, se les asocian diferentes gráficas. Por ejemplo: el ARN *6754* muestra el mayor número de estructuras relevantes en común con las estructuras presentes en el conjunto A cuando se analiza con el programa NUPACK Web, lo que establece que es más probable de ser un ARN-lnc. Sin embargo, el mismo ARN no fue identificado como ARN-lnc cuando se usa RNAfold WebServer o rna-state-inf, lo que subraya la influencia del modelo para el plegamiento de la estructura secundaria y los parámetros

de energía libre que emplean para generar su representación gráfica, influyendo así en el resultado de la predicción. Con esto en mente, una evaluación más detallada de la sensibilidad del método propuesto en distintos programas de plegamiento se podría abordar como trabajo futuro.

5.2. Caso: ARNs-lnc de *Homo sapiens*

Dado los buenos resultados obtenidos al clasificar ARNs-lnc provenientes de la levadura *S. cerevisiae*, mediante la metodología expuesta en la sección 3.3, el algoritmo propuesto en 4.1 y la búsqueda usando subconjuntos como en la sección 5.1.2. Se exploró el clasificar ARNs-lnc provenientes de *Homo sapiens*. Para este fin, se establecieron tres conjuntos de ARNs, todos pertenecientes a dicho organismo. En el Apéndice D se encuentran los datos de las secuencias analizadas, así como el número de nucleótidos que las conforman. A saber:

• **Grupo control** $\bar{A}$: constó de 50 secuencias aleatorias registradas en la literatura como ARNs-lnc.

• **Grupo control** $\bar{B}$: constó de 50 secuencias aleatorias registradas en la literatura que no pertenecen al conjunto de ARNs-lnc, ARNs-otros.

• **Grupo de prueba** $\bar{C}$: constó de 20 secuencias de ARNs también conocidos y seleccionados de manera aleatoria, las 10 primeras de ellas ARNs-lnc, mientras que las 10 restantes no pertenecen al conjunto de ARNs-lnc.

De manera similar a lo expuesto en la sección 5.1.2, dado que los conjuntos A y B para analizar constan de 50 elementos, para validar el análisis, se seleccionaron de manera aleatoria 30 subconjuntos con 35 elementos del conjunto A, A_i y 30 subconjuntos de solo 10 elementos de A que fueron parte del conjunto de prueba C, CA_i, donde $i = 1,2,\ldots,30$. Cabe señalar que los conjuntos A_i y CA_i son disjuntos. De manera análoga, se obtuvieron los 30 subconjuntos con 35 elementos del conjunto B, B_j y CB_j, donde $j = 1,2,\ldots,30$. El conjunto C está formado por un CA_i y un CB_j. Además, el valor para los parámetros l_4 y l_5 son diferentes a los tomados en el caso de analizar los ARNs-lnc de la levadura *S. cerevisiae*, cuyos conjuntos A y B poseían 18 elementos. A saber:

$$l_4 \geq \frac{|A|}{3} = \frac{|50|}{3} = 16.6 \quad \text{y} \quad l_5 \leq \frac{|B|}{4} = \frac{|50|}{4} = 12.5$$

Así, para este caso se tomaron $l_4 = 19$ y $l_5 = 8$, en vez de 7 y 3. Del mismo modo a lo expuesto en la sección 5.1.2, se definió K como el vector que realizó el seguimiento de los ARNs-lnc detectados en C. Luego, para obtener el vector K se iteraron los i-ésimos y j-ésimos elementos de los conjuntos A_i, B_j, y se fijó un valor para el conjunto C con $i = 3$ y $j = 5$:

$K = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$

para i en el rango (30):

  para j en el rango (30):

    $K = K + \mathcal{A}(A_i, B_j, CA_3 + CB_5, 22, 34, 2, 19, 8)$

Ahora bien, si se analiza solo el elemento con el mayor número de veces asignado como

ARN-lnc y este se asocia a un nuevo vector, V_R, en el cual se asigne el número 1 para indicar que el elemento fue detectado como un ARN-lnc el mayor número de veces. Por ejemplo, en el caso $V_C = [70,406,597,149,0,0]$, se tomará como el ARN-lnc con el mayor número de veces detectado el elemento tres de V_C obteniendo así el vector $V_R = [0,0,1,0,0,0]$. Así, por ejemplo, si ahora se pone a prueba el algoritmo 500 veces y se varía el conjunto $CA_{3j} + CB_{5j}$, para $j = 1,\ldots,500$, y por cada iteración se suma el vector obtenido V_R se tendrá como resultado el vector:

$$V_R = [216,79,75,40,21,22,12,14,11,8,0,0,0,0,1,0,0,0,1,0],$$

el cual indica que 498 veces las secuencias detectadas están dentro del conjunto de los ARNs-lnc, y al mismo tiempo, que 2 veces los elementos del conjunto de los ARNs que no pertenecen a los ARNs-lnc fueron detectados como ARNs-lnc (falsos positivos). En este caso, el 99.6 % de las veces fueron detectados correctamente como probables ARNs-lnc, en contraste con el 0.4 % de las veces en el que no lo fue. Los resultados se muestran en la Figura 5.6, donde: A3_1, A3_2, ... , A3_10 representan elementos del conjunto A, y B5_1, B5_2, ..., B5_10 representan elementos del conjunto B, que cambian en cada iteración, pero se analiza su aparición de manera acumulativa.

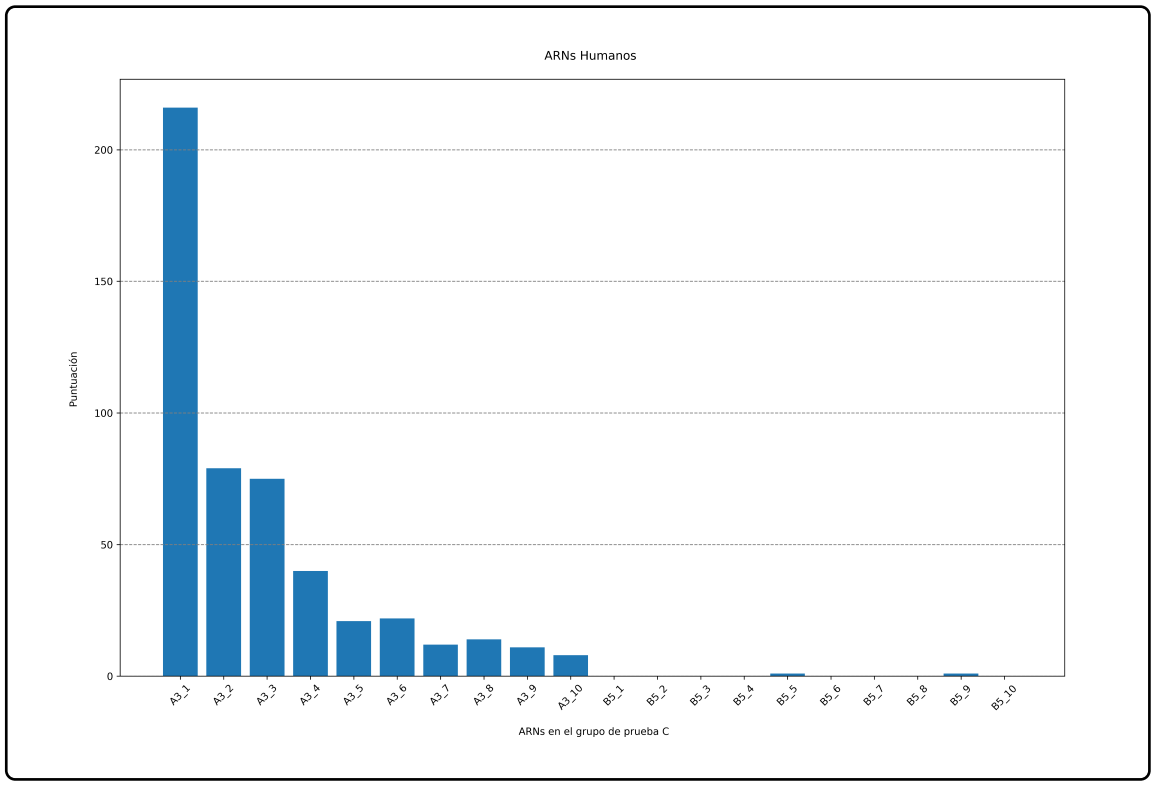


Figura 5.6: Los resultados del algoritmo después de 500 iteraciones: con $C = CA_{3j} + CB_{5j}$.

Por otro lado, si se pone a prueba el algoritmo 500 veces y se iteran los i-ésimos y j-ésimos elementos de los conjuntos A_i, B_j, de $1,\ldots,500$. Pero ahora, fijando el conjunto C como el grupo de prueba C original y por cada iteración se suma el vector obtenido V_R se

tendrá como resultado:

$$V_R = [0,0,0,6,0,450,3,0,0,0,1,1,0,0,0,39,0,0,0,0]$$

En la Figura 5.7 se muestran los resultados. Lo cual sugiere que el ARN *lnc-INTS14-1:2* es, en este caso, el más probable de ser un ARN-lnc.

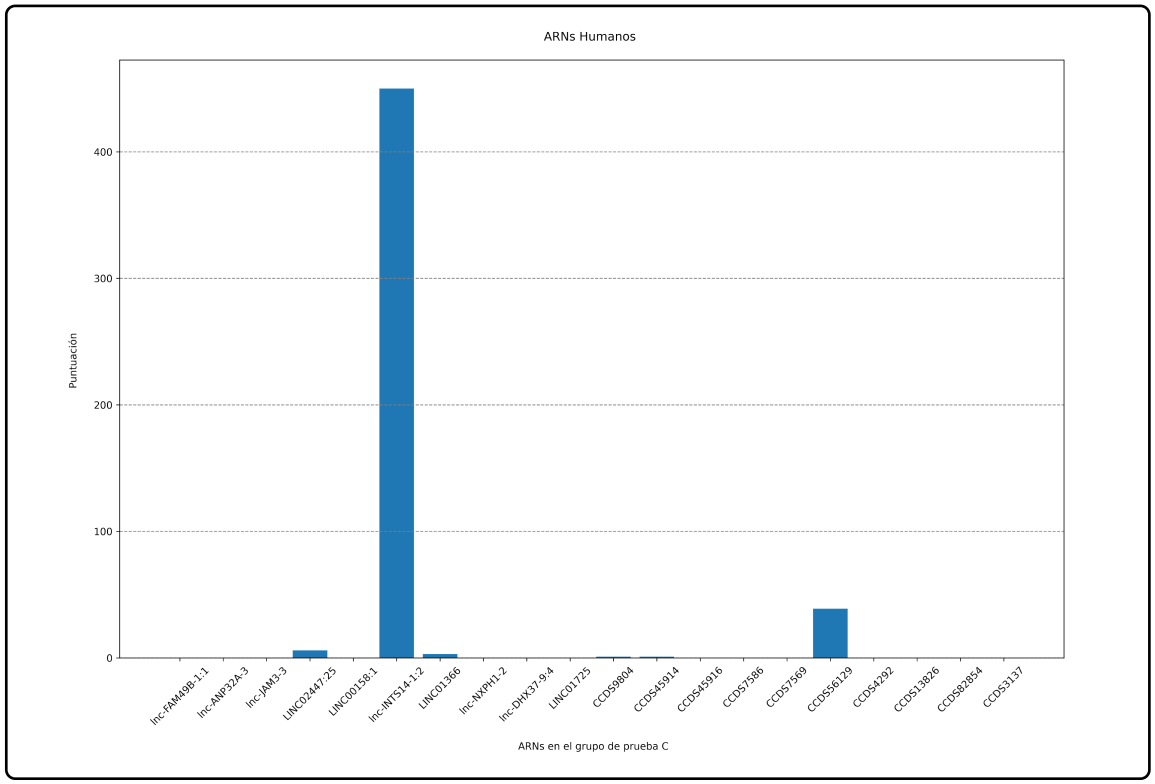


Figura 5.7: Los resultados del algoritmo después de 500 iteraciones: con los datos del conjunto de prueba *C*.

El hecho de que el ARN *lnc-INTS14-1:2* sea detectado como un probable ARN-lnc es una buena detección, dado que los primeros 10 elementos del conjunto de prueba *C* son ARNs-lnc. En resumen: 459 veces, de las 500, tres de los primeros 10 ARNs se detectaron correctamente como ARNs-lnc, mientras que 41 veces se tuvieron falsos positivos, es decir, el 91.8 % de las veces el algoritmo identificó correctamente un ARN-lnc, en contraste con el 8.2 % en el que no lo hizo. Como se puede observar, la metodología y el algoritmo propuestos en esta tesis son útiles también en la identificación de ARNs-lnc provenientes de *Homo sapiens*. Con esto en mente, el analizar ARNs-lnc aún no clasificados o asociados a alguna otra característica funcional se podría abordar como trabajo futuro.

Capítulo 6

El código del algoritmo

Los conjuntos de datos analizados, el algoritmo propuesto y un breve manual para su uso pueden ser consultados en el siguiente repositorio de GitHub (versión en inglés, consultado el 25 de noviembre de 2025):

https://github.com/Dave-HG/A-graph-based-algorithm-for-detecting-lncRNAs-through-RNA-secondary-structure-analysis

El algoritmo está conformado por dos partes:

En la primera parte, dado los grupos control $\bar{A}$ y $\bar{B}$, y el grupo de prueba $\bar{C}$, de secuencias de ARN en *NPP* se transforman a *NPPS*, puede encontrar el código correspondiente en la sección 6.1.

En la segunda parte, el algoritmo analiza los grupos control $\bar{A}$ y $\bar{B}$ para determinar si es posible que un ARN del grupo de prueba $\bar{C}$ sea un ARN-lnc, puede encontrar el código correspondiente en la sección 6.2.

6.1. De Notación Punto-Paréntesis a Notación Punto-Paréntesis Simplificada

```
########################################################################
##############       INICIAN FUNCIONES                  ##############

########################################################################
###  Dada la NPP se obtiene un vector de numeros que la codifican  ###
########################################################################
def dbn2vector(P):
    p = len(P)
    PairsP = []
    for i in range(p):
        if P[i] == '.':
            PairsP.append([i, i])
        elif P[i] == '(':
```

```
            PairsP.append([i, 1])
        elif P[i] == ')':
            PairsP.append([i, -1])

    for r in range(p):
        if PairsP[r][1] == -1:
           for s in range(r):
                if PairsP[r-s][1] == 1:
                    PairsP[r][1] = PairsP[r-s][0]
                    PairsP[r-s][1] = PairsP[r][0]
                    break
    return (PairsP)

##################################################
##   Dado el vector numerico se obtiene la NPP   ##
##################################################
def vector2dbn(E):
    e = len(E)
    G = ''
    for j in range(e):
        if E[j][0] == E[j][1]:
            G = G+'.'
        elif E[j][0] < E[j][1]:
            G = G+'('
        elif E[j][0] > E[j][1]:
            G = G+')'
    return G

##########################################################
## Se buscan tallos con un par de bases y se eliminan ##
##########################################################
def remstep(E):
    e = len(E)
    conte = 0
    for g in range(e-2):
        if E[g][0] == E[g][1] and E[g+1][0] != E[g+1][1] and E[g+2][0] == E[g+2][1]:
            p = E[g+1][1]
            conte = conte+1
    for h in range(conte):
        g = 0
        while g < e-2:
            if E[g][0] == E[g][1] and E[g+1][0] != E[g+1][1] and E[g+2][0] == E[g+2][1]:
                p = E[g+1][1]
                if E[p-1][0] == E[p-1][1] and E[p][0] != E[p][1] and E[p+1][0] == E[p
    +1][1]:
                    E[g+1][1] = E[g+1][0]
                    E[p][1] = E[p][0]
                    break
            g = g+1
    return E

#########################################################################
# Se eliminan las protuberancias sin dos pares de bases consecutivos #
#########################################################################
def rembulge(D):
    E = D
    n = len(E)
    cont = 0
    for g in range(n-2):  # Este es un punto E[g+1][0]==E[g+1][1].
        if E[g+1][0] == E[g+1][1] and D[g][0] != D[g][1] and D[g+2][0] != D[g+2][1]:
            if abs(D[g][1]-D[g+2][1]) < 3:
                cont = cont+1
    for h in range(cont):
        n = len(D)
        i = 0
        while i < n:
            if D[i][0] != D[i][1] and D[i+1][0] == D[i+1][1] and D[i+2][0] != D[i+2][1]:
                if abs(D[i][1]-D[i+2][1]) == 1:
```

```
                    for k in range(n):
                        if D[k][0] > D[i+1][0]:
                            D[k][0] = D[k][0]-1
                    for k in range(n):
                        if D[k][1] > D[i+1][1]:
                            D[k][1] = D[k][1]-1
                    D.remove(D[i+1])
                    break
                if abs(D[i][1]-D[i+2][1]) == 2:
                    g=D[i+2][1]+1
                    D[g][1] = D[i+1][0]
                    D[i+1][1]=D[g][0]
            i = i+1
    return D

############################################################
# Se agrega un punto en las uniones donde sea necesario #
############################################################
def addpoint(E):
  n=len(E)
  cont3=0
  for g in range(n-2):
    if E[g][0]!=E[g][1] and E[g+1][0]!=E[g+1][1]:
        if abs(E[g][1]-E[g+1][1])>1:
          cont3=cont3+1
  for h in range(cont3):
    g=0
    while g<n:
      if E[g][0]!=E[g][1] and E[g+1][0]!=E[g+1][1]:
          if abs(E[g][1]-E[g+1][1])>1:
            for k in range(n):
              if E[k][0]>E[g][0]:
                E[k][0]=E[k][0]+1
            for k in range(n):
              if E[k][1]>E[g][0]:
                E[k][1]=E[k][1]+1
            E.insert(E[g][0]+1, [E[g][0]+1,E[g][0]+1])
            break
      g=g+1
  return E

###############            TERMINAN FUNCIONES              ###############
#########################################################################


#########################################################################
##############             EMPIEZA ALGORITMO                ##############

#############################################################################
#######     Se agregan los tres conjuntos en NPP: A=AA0, B=BB0 y C=CC0      #######
#############################################################################

#AA0 es el conjunto de ARNs-lnc en NPP (Notacion Punto-Parentesis).
NOMLNC=['ICR1','RME2','RME3','IRT1','TLC1','PWR1','RUF5-1','RUF21','ETS1-1','SRG1',
'RUF22','RUF20','ITS1-1','RUF23','ITS2-1','ETS2-1','RNA170','ZOD1']

a0='......((((  ***  ))))......'
a1='.(((((..(  ***  ...)))))).'
a2='.....(((((  ***  ..........'
a4='((((..((((.  ***  ....)))))..'
a5='.(((((((..  ***  )).........'
a6='((((......  ***  ....)))))..'
a7='........((  ***  ..)))).....'
a8='.(((((((((  ***  ..)))))))..'
a9='....(((...  ***  ...).)))))'
a10='..(((((((  ***  )).........'
a11='.(((((((.  ***  ..........'
a12='..((((((.  ***  ..........'
```

```
a13='(((((((..  ***  ((....))).'
a14='........(  ***  ))))......'
a15='....(((((  ***  .)))))))...'
a16='.........  ***  ..........'
a17='(((.(((((  ***  ..........'

AA0=[a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17]


#BB0 es el conjunto de otros ARNs que no son ARNs-lnc en NPP (Notacion Punto-Parentesis).
NOMCOD=['15S ribosomal','LSR1','Telomerasa','SNR86','SNR30','Small nucleolar SNR30','SNR190'
    ,'SNR84','Small nucleolar SNR84','RPM1','SNR17B','Nuclear RNASE P','SNR42','NME1','U3',
    'Small nucleolar U3','RNASE MRP','SNR83']

b0='..........  ***  )))))))...'
b1='((((((.(((  ***  ..........'
b2='(((((((...  ***  (....))...'
b3='.....(((..  ***  ))).......'
b4='.......(((  ***  )))))))))).'
b5='.......(((  ***  ))))))....'
b6='..((((..((  ***  )))))))))).'
b7='.(((((((((  ***  ))).)))...'
b8='.(((((((((  ***  )))))).....'
b9='..........  ***  ......))).'
b10='(((...(((  ***  )).)))....'
b11='.(((((((((  ***  .)))))))).'
b12='......(((  ***  ))))......'
b13='..(((((((  ***  ))).......'
b14='.........  ***  ))...)))))'
b15='...((.(((  ***  ).))......'
b16='.(((((((((  ***  ))))).)))))'
b17='..(((((((  ***  )))))))...'

BB0=[b0,b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15,b16,b17]


#CC0 es el conjunto de posibles ARNs-lnc en NPP (Notacion Punto-Parentesis).
NOMPOSIBLES=['KAP123','GRE2','EMC11', '1477','6754','12189']

c0='..(((.....  ***  ...)))))))'
c1='.(((((((..  ***  ))).......'
c2='....(((((.  ***  ..........'
c3='..........  ***  ))).......'
c4='((.......(  ***  ..........'
c5='(((((((...  ***  ))))......'

CC0=[c0,c1,c2,c3,c4,c5]

######################################################################
########       Se obtiene la NPPS de: AA0, BB0 y CC0       ########
######################################################################


#DD0=AA0    # Se usa para determinar la NPPS del conjunto A (ARNs-lnc).
#DD0=BB0    # Se usa para determinar la NPPS del conjunto B (ARNs-otros).
DD0=CC0     # Se usa para determinar la NPPS del conjunto C (posibles ARNs-lnc).

w = len(DD0)
AA1 = []
for l in range(w):
    AA1.append('...' + DD0[l] + '...')

Reduc=[]


print("\n=============================================")
print('Espere por favor, transformando la NPP a NPPS:')
print("=============================================\n")
```

```
for j in range(w):
      # Aplicando las reglas de Gan a cada cadena en NPP.
      P = AA1[j]
      D = dbn2vector(P)
      H1 = remstep(D)
      H2 = rembulge(H1)
      H3=addpoint(H2)
      H4=remstep(H3)
      K4=vector2dbn(H4)

      # Cadenas con los mismos caracteres consecutivos se reducen a un solo caracter.
      aaa=len(K4)
      B='.'
      for k in range(aaa-1):
          if K4[k]!=K4[k+1]:
              B=B+K4[k+1]
      print(j,B)  # "j" es el numero de la secuencia transformada y "B" su cadena en NPPS
      print()
      Reduc.append(B)
pass

##############             TERMINA ALGORITMO               ##############
#########################################################################
```

6.2. Bootstrapping

```
#########################################################################
################             IMPORTAR LIBRERIAS              ################
#########################################################################
import xlwt
import numpy as np
import random


#########################################################################
################             EMPIEZAN FUNCIONES              ################

#########################################################################
# compare(seq1, seq2, inf) compara los vectores seq1 y seq2; cadenas en NPPS, #
# para establecer el vector subst; subcadenas compartidas entre ellos, de lon-#
# gitud >= inf.                                                               #
#########################################################################
def compare(seq1,seq2,inf):
    a=seq1
    b=seq2
    m=len(b)     # "m" es la longitud maxima de la cadena a buscar.
    minor=inf
    subst=[]     # "subst" guarda las subcadenas encontradas, pero sin repeticiones.
    start=[]     # "start" es el vector que indica la posicion de inicio de la subcadena.
    end=[]       # "end"   es el vector que indica la posicion final de la subcadena.

    ''' Busca las subcadenas de longitud mayor o igual que "minor". '''
    for u in range(0,m-minor+1):   # "u" es el inicio de la subcadena a buscar.
                                   # Empieza en longitud m y termina en longitud "minor".
        for v in range(0,u+1):
            d=b[v:v+(m-u)]         # En cada iteracion de "u", la subcadena tiene
                                   # la longitud de "m-u".
            m1=len(d)
            a2=a
            c=a2.find(d)           # "c" es la primera posicion donde "d" es encontrada.
                                   # Si c=-1 entonces "d" no esta en "a2".
            t=c                    # "t" guardara la posicion inicial,
                                   # mientras que "c+m1" guardara la posicion final.
            if c>-1:
```

```
                if t+1 not in start and t+1+m1 not in end:
                    if d not in subst:
                        subst.append(d)
                    pass
                start.append(t+1)  # Agrega posicion inicial de "d".
                end.append(t+1+m1) # Agrega posicion final de "d".
    return subst

#############################################################################
##  stnorep(subst) analiza el vector subst y selecciona las cadenas que    ##
##  no se repiten para establecer el vector comstnotrep; subcadenas com-  ##
##  partidas sin repeticion.                                               ##
#############################################################################
def stnotrep(subst):
    comstnorep =[]  # Guarda todas las subcadenas compartidas sin repeticion.
    s=len(subst)
    for j in range(s):
        if subst[j] not in comstnorep:
            comstnorep.append(subst[j])
    return comstnorep

#############################################################################
##  matrix(comstnorep, set) cuantifica las veces que el elemento          ##
##  comstnorep[j] esta presente en set[i].                                ##
#############################################################################
def matrix(comstnorep,Set):
    data=[]      # "data" sera el vector de longitud "a x s" con el numero
                 # de veces "cont" que la subcadena comstnorep[j] esta
                 # presente en la subcadena Set[i].
    a=len(Set)
    s=len(comstnorep)
    for i in range(a):
        for j in range(s):
            cont=Set[i].count(comstnorep[j])
            data.append(cont)
    Data=np.array(data).reshape(a,s)  # Transforma el vector "data"
                                      # en una matriz de "a x s".
    return Data

################              TERMINAN FUNCIONES              ################
#############################################################################

#############################################################################
##############    Generar el .xls con los resultados        ##############
#############################################################################
# Crear el libro y la hoja.
libro = xlwt.Workbook()
hoja = libro.add_sheet('Sheet 1')

# Escribir los datos en celdas.
hoja.write(0, 0, 'Ronda j')
hoja.write(0, 1, 'Vector K')
hoja.write(0, 2, 'Vector V_R en ronda j')

#############################################################################
##############             EMPIEZA ALGORITMO               ##############

#############################################################################
#####    Se agregan los tres conjuntos en NPPS: A=AA0, B=BB0 y C=CC0      #####
#############################################################################

a0='.(.(.(.(.(  ***  (.(.(.).).'
a1='.(.(.).(.(  ***  (.).).).).'
a2='.(.(.(.(.)  ***  ).).(.).).'
a3='.(.(.(.(.(  ***  (.).).).).'
a4='.(.(.(.(.(  ***  ).).).).).'
a5='.(.(.(.(.(  ***  ).).).).).'
a6='.(.(.(.).(  ***  (.).).).).'
```

```
a7='.(.).(.(.)  ***  (.).).).).'
a8='.(.(.(.).)  ***  (.).).).).'
a9='.(.).(.(.)  ***  ).).).(.).'
a10='.(.).(.(.  ***  ).(.).).).'
a11='.(.(.(.(.  ***  ).).).).).'
a12='.(.).(.(.  ***  ).(.).).).'
a13='.(.).(.(.  ***  ).).).(.).'
a14='.(.(.(.(.  ***  ).).).).).'
a15='.(.).(.(.  ***  ).(.).(.).'
a16='.(.(.(.).  ***  ).).(.).).'
a17='.(.(.).).'

AA0=[a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17]

b0='.(.(.(.(.(  ***  (.).).).).).'
b1='.(.(.(.).)  ***  (.).).).).'
b2='.(.(.(.(.(  ***  ).).).(.).'
b3='.(.(.(.(.(  ***  ).(.(.).).'
b4='.(.(.(.(.(  ***  (.).).).).'
b5='.(.(.(.(.(  ***  ).).).).).'
b6='.(.(.(.(.(  ***  ).).).(.).'
b7='.(.(.(.(.)  ***  ).).).).).'
b8='.(.(.(.(.)  ***  ).).(.).).'
b9='.(.(.(.(.)  ***  ).(.).).).'
b10='.(.(.(.(.  ***  (.).).).).'
b11='.(.(.(.(.  ***  (.).(.).).'
b12='.(.(.(.(.  ***  (.).).).).'
b13='.(.(.(.(.  ***  (.(.).).).'
b14='.(.).(.(.  ***  ).).).).).'
b15='.(.(.(.(.  ***  (.(.).).).'
b16='.(.(.(.(.  ***  (.(.).).).'
b17='.(.(.(.(.  ***  (.(.).).).'

BB0=[b0,b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15,b16,b17]


# Posibles ARNs-lnc.
NOMPOSSIBLES=['KAP123','GRE2','EMC11','1477','6754','12189']

c0='.(.).(.(.(  ***  ).).).).'
c1='.(.(.(.(.)  ***  ).(.(.).).'
c2='.(.).(.(.)  ***  ).(.).).).'
c3='.(.(.).).(  ***  (.).).).).'
c4='.(.(.(.(.)  ***  ).).).).).'
c5='.(.(.(.(.)  ***  (.).).(.).'

CC0=[c0,c1,c2,c3,c4,c5] #Los primeros tres no pertenecen a los ARNs-lnc.

print("\n=====================================================")
print('Espere por favor, analizando los conjuntos de datos:')
print("=====================================================\n")

DDS1=AA0   # "DDS1" es el vector de A con las secuencias en NPPS.
DDS2=BB0   # "DDS2" es el vector de B con las secuencias en NPPS.

############################
# Restriccion:
############################
l0=16      # "l0" es la longitud minima de la subcadena repetida que sera buscada.

#########################################################################
##########   Compara las subcadenas en NPPS de cada conjunto   ##########
#########################################################################
lng = len(AA0) # Note que len(AA0)=len(BB0).
shseq21=[]     # Guarda todas las secuencias compartidas de A con repeticiones.
shseq22=[]     # Guarda todas las secuencias compartidas de B con repeticiones.

for im in range(lng):
```

```
    for jm in range(lng-im-1):
        f1=compare(DDS1[im],DDS1[im+jm+1],l0)
        shseq21.extend(f1)    # Agrega la subcadena compartida de A de longitud >= l0.
        f2=compare(DDS2[im],DDS2[im+jm+1],l0)
        shseq22.extend(f2)    # Agrega la subcadena compartida de B de longitud >= l0.

# Remover las subcadenas repetidas en el conjunto de subcadenas compartidas:

SNR21=stnotrep(shseq21)  # Secuencias repetidas de A.
SNR22=stnotrep(shseq22)  # Secuencias repetidas de B.
r1=len(SNR21)
r2=len(SNR22)

#########################################################################
##  Tomando subcadenas de longitud entre l1 y l2 con l0 < l1 < l2   ###
#########################################################################

############################
# Restriccion:
############################
l1=22
l2=34
############################

SSRRA=[]                  # SSRRA son las cadenas con longitud entre l1 y l2 de SNR21.
for ll in range(r1):
  long=len(SNR21[ll])
  if l1 <= long <= l2:
    SSRRA.append(SNR21[ll])

SSRRB=[]                  # SSRRB son las cadenas con longitud entre l1 y l2 de SNR22.
for ll in range(r2):
  long=len(SNR22[ll])
  if l1 <= long <= l2:
    SSRRB.append(SNR22[ll])

#########################################################################
# Se toman aleatoriamente (quantity=30) subconjuntos de (numelements=12)   #
# elementos de A, Ai, y 30 subconjnutos de (to_C=3) elementos de A, CAi.   #
# Los conjuntos Ai y CAi son disjuntos.                                    #
# Similarmente se obtienen los subconjuntos Bj y CBj para j=1, 2, ..., 30  #
#########################################################################

################################
# Generar conjuntos aleatorios:
################################
quantity=30
numelements=12
to_C=3
################################

print('Usando',
quantity,'subconjuntos con',numelements,'elementos de A,',
quantity,'subconjuntos con',numelements, 'elementos de  B y',
quantity,'subconjuntos con',2*to_C, 'elementos', to_C, 'de A y', to_C, 'de B.')
print()

hits=0      # "hits" mantiene el seguimiento del numero de las veces
            # que se detecto correctamenete un ARN-lnc.
p0=0        # "pj" mantiene el seguimiento del numero de las veces
            # que el elemnto "j" de "CC0" es detectado como un ARN-lnc.
p1=0
p2=0
p3=0
p4=0
p5=0

n = len(AA0)
```

```
tod = list(range(n))
prom=0


##############
trials=500              # Las veces que sera puesto a prueba el algoritmo:
                        # trials=500 $\mathcal{A}(A_i,B_j,CA_3+CB_5)$.
##############

for w in range(trials):  # Variar RC=X[w] en cada iteracion.

    ###########################################################################
    ##  Generar aleatoriamente la lista de numeros, la cual sera usada       ##
    ##  para obtener $A_i,CA_i, B_j,CB_j$                                    ##
    ###########################################################################

    ranA=[]
    ranB=[]
    raaC=[]
    rabC=[]
    for k in range(quantity):
      raA=[]
      raB=[]
      raC=[]
      rbC=[]

      while len(raA)<numelements+to_C:
        af=random.choice(tod)
        if af not in raA:
          raA.append(af)
      ranA.append(raA[0:numelements])
      raaC.append(raA[numelements:numelements+to_C])

      while len(raB)<numelements+to_C:
        ag=random.choice(tod)
        if ag not in raB:
          raB.append(ag)
      ranB.append(raB[0:numelements])
      rabC.append(raB[numelements:numelements+to_C])

    RA=ranA   # "quantity" subconjuntos con "numelements" elementos de A.
    RB=ranB   # "quantity" subconjuntos con "numelements" elementos de B.
    RaC=raaC  # "quantity" subconjuntos con "to_C" elementos de A.
    RbC=rabC  # "quantity" subconjuntos con "to_C" elementos de B.

    n24=0     # "n_j" cuantifica el numero de veces que el $j$-esimo ARN del conjunto $C$
              # es detectado como un ARN-lnc.
    n25=0
    n26=0
    n27=0
    n28=0
    n29=0

    ###########################################################################
    ########     EL ALGORITMO $\mathcal{A}(A_i,B_j,CA_3+CB_5)$         ########
    for x in range(quantity):
      for y in range(quantity):

        ###########################################################################
        ## Obtener los conjuntos: $A_x,CA_x, B_y,CB_y$ y $D=A_x + B_y+,CA_x, + CB_y$ ##
        ###########################################################################
        RAA0=[]
        RBB0=[]
        RCC0=[]
        for j in range(numelements):
          RAA0.append(AA0[RA[x][j]])
          RBB0.append(BB0[RB[y][j]])
```

```
RC=RaC[3]+RbC[5]    # Se fija el tercer conjunto como: $CA_3+CB_5$

for j in range(to_C):
  RCC0.append(AA0[RC[j]])
for k in range(to_C):
  RCC0.append(BB0[RC[2+k]])

#####################################
# AQUI SE ELIGEN CUAL CC0 SERA USADO #
#####################################
#DD0=RAA0+RBB0+RCC0 # Use este conjunto como DD0 para usar
                    # CC0 con el bootstrapping.
DD0=RAA0+RBB0+CC0   # Use este conjunto como DD0 para usar
                    # CC0 como el conjunto de posibles ARNs-lnc.

#####################################################################
# Calcular las matrices: $M_{AA}$, $M_{AB}$, $M_{BB}$ y  $M_{BA}$ #
#####################################################################

SSRRAenA=matrix(SSRRA,RAA0)
SSRRAenB=matrix(SSRRA,RBB0)
SSRRBenB=matrix(SSRRB,RBB0)
SSRRBenA=matrix(SSRRB,RAA0)

# Calcula los vectores: $C_{AA}$, $C_{AB}$, $C_{BB}$ y $C_{BA}$.
SumColAnA=np.sum(SSRRAenA, axis=0)
SumColAnB=np.sum(SSRRAenB, axis=0)
SumColBnB=np.sum(SSRRBenB, axis=0)
SumColBnA=np.sum(SSRRBenA, axis=0)

# Determinar: $C_{AA}-C_{AB}$ y $C_{BB}-C_{BA}$.
difSSRRA=SumColAnA-SumColAnB
difSSRRB=SumColBnB-SumColBnA

############################
# Restriccion:
############################
l3=2
############################

# Determinar: $S^{'}_{A}$ y $S^{'}_{B}$.
L=len(SumColAnA)
NEWSSRRA=[]          # Es el nuevo vector $S^{'}_{A}$.
for j in range(L):
  if  difSSRRA[j]>=l3:
      NEWSSRRA.append(SSRRA[j])

LL=len(SumColBnB)
NEWSSRRB=[]          # Es el nuevo vector $S^{'}_{B}$.
for j in range(LL):
  if  difSSRRB[j]>=l3:
      NEWSSRRB.append(SSRRB[j])

# Calcular las matrices: $M^{'}_{AA}$, $M^{'}_{BB}$, $M^{'}_{AD}$, y $M^{'}_{BD}$
NEWSSRRAenA=matrix(NEWSSRRA,RAA0)
NEWSSRRBenB=matrix(NEWSSRRB,RBB0)
NEWSSRRAenC=matrix(NEWSSRRA,DD0)
NEWSSRRBenC=matrix(NEWSSRRB,DD0)

# Calcular los vectores: $R_{AA}$, $R_{BB}$, $R_{AD}$ y $R_{BD}$.
sumrowNEWSSRRAenA=np.sum(NEWSSRRAenA, axis=1)
sumrowNEWSSRRBenB=np.sum(NEWSSRRBenB, axis=1)
sumrowNEWSSRRAenC=np.sum(NEWSSRRAenC, axis=1)
sumrowNEWSSRRBenC=np.sum(NEWSSRRBenC, axis=1)

# Calcular el primer cuartil en A, $Q1_A$, y en B, $Q1_B$.
q1a=np.quantile(sumrowNEWSSRRAenA, .25)
q1b=np.quantile(sumrowNEWSSRRBenB, .25)
```

```
    # Determinar si la $i$-esima entrada de $R_{ADi}$ y $R_{BDi}$ satisface la
restriccion:
    # $R_{ADi}>0.5*Q1_A$ $R_{BDi}<1.5*Q1_B$.
    cc=len(DD0)
    vector=[]     # Guarda los ARN de $C$ que satisacen la restriccion.
    vectorA=[]    # Guarda los ARN de $Ax$ que satisacen la restriccion.
    vectorB=[]    # Guarda los ARN de $By$ que satisacen la restriccion.
    for kk in range(cc):
      if ((q1a *.5)<= sumrowNEWSSRRAenC[kk]) and (sumrowNEWSSRRBenC[kk] <= (q1b*1.5)):
        vector.append(kk)
        if (kk<= len(AA0)):
          vectorA.append(kk)
        if ( len(AA0)+1<=kk<=  len(AA0)+ len(AA0)):
          vectorB.append(kk)

    # Determinar si los numeros $N_A=len(vectorA)$ y $N_B=len(vectorB)$
    # de elementos de $A$ y $B$, satisfacen la restriccion:
    # $N_A\geq l_4=7$ and $N_B\leq l_5=3$.

    ############################
    # Restriction:
    ############################
    l4=7
    l5=3
    ############################

    if (l4<= len(vectorA)) and (len(vectorB) <= l5):
      # "n_j" cuantifica el numero de veces que el $j$-esimo ARN del conjunto $C$
      # es detectado como un ARN-lnc.
      if 24 in vector:
          n24=n24+1
      if 25 in vector:
          n25=n25+1
      if 26 in vector:
          n26=n26+1
      if 27 in vector:
          n27=n27+1
      if 28 in vector:
          n28=n28+1
      if 29 in vector:
          n29=n29+1

list=[n24,n25,n26,n27,n28,n29]        # El conjunto $K$.
aaa=np.amax([n24,n25,n26,n27,n28,n29])
bbb=np.amax([n24,n25,n26])
if aaa==bbb:
    hits=hits+1

position=list.index(aaa)
if 0==position:
    p0=p0+1
if 1==position:
    p1=p1+1
if 2==position:
    p2=p2+1
if 3==position:
    p3=p3+1
if 4==position:
    p4=p4+1
if 5==position:
    p5=p5+1

V_R=[p0,p1,p2,p3,p4,p5]               # El conjunto $V_R$.
print('Resultados de la ronda',w+1,':',list, '. Aqui, es detectada como ARN-lnc la
secuencia numero:', position+1)
print('Resultados acumulativos',V_R)
print()
```

```
    # Guardar los resultados en el libro.
    hoja.write(w+1, 0, w+1)
    hoja.write(w+1, 1, str([n24,n25,n26,n27,n28,n29]))
    hoja.write(w+1, 2, str([p0,p1,p2,p3,p4,p5]))

libro.save('Resultados.xls')

##############             TERMINA ALGORITMO                ##############
################################################################################
```

6.3. Para usar el algoritmo

Para usar el algoritmo, por ejemplo para el caso de ARNs-lnc de *S. cerevisiae*, en su versión en Inglés, ingresar al repositorio de GitHub:

https://github.com/Dave-HG/A-graph-based-algorithm-for
-detecting-lncRNAs-through-RNA-secondary-structure-analysis

Seleccionar la carpeta *S. cerevisiae*.

Descargar todos los archivos presentes, en este caso: From_DBN_to_SDBN.py, Bootstrapping Algorithm A.py, Data Nupack.xls, Data RNAfold.xls, Data RNAstate.xls. Y seguir las siguientes instrucciones (mostradas en el archivo README.md):

1. Usar un programa de plegamiento, como NUPACK Web, para dados los datos de las secuencias de ARN en los conjuntos $\bar{A}$, $\bar{B}$ y $\bar{C}$ obtener sus correspondientes conjuntos en *NPP*: $\hat{A}$, $\hat{B}$ y $\hat{C}$. Puede usar las secuencias preestablecidas de los archivos: Data Nupack.xls, Data RNAfold.xls o Data RNAstate.xls.

2. Ingresar los conjuntos en *NPP* en el archivo From_DBN_to_SDBN.py (lineas 127-187) para obtener los conjuntos A, B y C en *NPPS*.

3. Ingresar los conjuntos en *NPPS* en el archivo Bootstrapping Algorithm A.py (líneas 74-127) para obtener el vector V_R con los resultados.

Los valores de los parámetros de l_1 a l_5 y $trials$ pueden ser cambiados en:

l_1 en línea 195.
l_2 en línea 196.
l_3 en línea 353.
l_4 en línea 402.
l_5 en línea 403.
$trials$ en línea 249 (Número de veces que el algoritmo será ejecutado, por defecto trials=500).

4. Los resultados pueden ser consultados en el archivo Results.xls

De manera similar, para usar el algoritmo para el caso de ARNs-lnc de *Homo sapiens*, seleccionar la carpeta *Humans*, descargar los archivos presentes y seguir las instrucciones mostradas en el archivo README.md.

Capítulo 7

Conclusiones

Con la finalidad de analizar las similitudes estructurales entre secuencias de ARNs, se empleó el marco teórico de Gan *et al.* [4], el cual consiste en asociar una gráfica de árbol con raíz a una estructura secundaria de ARN, para con ello, estimar el repertorio de las estructuras secundarias del ARN y mostrar que las gráficas de árbol de ARNs conocidos representan un pequeño subconjunto de todos los posibles motivos estructurales. Por lo tanto, algunos de los motivos estructurales faltantes podrían representar nuevos ARNs diseñados. Basándonos en estas ideas, en esta tesis, se establecieron algunas reglas partiendo solo de una estructura secundaria en *NPP* asociarle su *NPPS*. Además, se propuso un algoritmo que permite encontrar cadenas compartidas entre elementos de dos conjuntos de ARNs, con el objetivo de establecer algunos parámetros que permitan diferenciar elementos entre ambos conjuntos de ARNs.

Dicho algoritmo fue probado con tres conjuntos de datos, los cuales fueron secuencias de ARNs pertenecientes al organismo *S. cerevisiae*, dos grupos control: $\bar{A}$ y $\bar{B}$, y el otro, el grupo de prueba $\bar{C}$, expuestos en el Cuadro 2.1. Resumiendo, se obtuvo un conjunto de cadenas que permiten determinar cuándo una secuencia de ARN es más probable que sea un ARN-lnc. Cabe remarcar que usando como conjunto $C = CA_{i_0} + CB_{j_0}$, para i_0 y j_0 fijo, el algoritmo detectó correctamente los ARNs que eran más probables de ser ARNs-lnc con porcentajes de precisión cercanos al 95 %, independientemente del programa de plegamiento que se emplee, en este caso NUPACK Web, RNAfold WebServer y rna-state-inf.

Como resultados, se estableció que los ARNs *6754* y *12189* son candidatos idóneos a ser ARNs-lnc. Es importante notar, que en 2020 Ana Novaičić *et al.* [47] validaron conforme a un experimento que el ARN *12189* es un ARN-lnc, lo cual sirve de soporte a la metodología propuesta. Así, dados los resultados, se sugiere que la metodología y el algoritmo propuestos pueden servir como una valiosa herramienta computacional para la identificación de ARNs-lnc pertenecientes a la levadura *S. cerevisiae*. Con ello, se establece que es muy factible analizar experimentalmente el ARN *6754*, con el fin de establecer si es un ARNs-lnc. Por otro lado, al emplear la metodología y el algoritmo para la identificación de ARNs-lnc pertenecientes a *Homo sapiens* los resultados indican que también es posible identificar ARNs-lnc de este organismo con un porcentaje de precisión cercano al 92 %.

Una diferencia importante entre el enfoque de Gan en el que exploran el panorama global de los motivos estructurales del ARN y el enfoque de Siyu *et al.* [12] que se basa en la composición de secuencias y en el aprendizaje automático para la predicción de ARNs-lnc, el trabajo presentado en esta tesis expone una estrategia fundamentalmente diferente. A saber, se definió un algoritmo basado en una serie de reglas que generan una representación en punto paréntesis de una secuencia de ARN e identifica patrones estructurales compartidos entre dichas secuencias; proporcionando con ello, una metodología directa que se centra en las estructuras principales para distinguir clases funcionales en el ARN y determinar nuevos candidatos idóneos a ser ARNs-lnc con un nivel de confianza del 95 %.

En resumen, utilizando la información de la estructura secundaria del ARN, se identificaron ARNs que comparten características estructurales con los ARNs-lnc reportados en la literatura, por ende, estos ARNs son más probables de que compartan alguna funcionalidad. Si bien, este análisis se centró en los ARNs-lnc, la metodología y el algoritmo propuestos pueden aplicarse para explorar otras funcionalidades de los ARNs, lo que abre una puerta a nuevas aplicaciones en la biología del ARN. Como un ejemplo de ello, estudios recientes han demostrado que las células tumorales secretan ARNs-lnc en fluidos biológicos humanos en forma de microvesículas, exosomas o complejos proteicos. Estas moléculas son secretadas en forma de ARNs-lnc circulares que permanecen en un estado estable y no son degradadas por las enzimas del ARN. Tales ARNs-lnc circulares pueden servir como biomarcadores en la detección de cáncer Hu *et al.* en [67] demuestran que los ARNs-lnc: *SPRY4-IT1*, *ANRIL* y *NEAT1* exhiben una alta especificidad y sensibilidad en el cáncer de pulmón de células no pequeñas (NSCLC), sugiriendo su uso potencial como nuevos marcadores en el diagnóstico de NSCLC. Similarmente, en el caso de cáncer de mama, Xu *et al.* en [68] comparan los valores ROC y AMC de ARNs-lnc con algunos marcadores tumorales séricos más comunes y demuestran que el ARN-lnc *RP11-445H22.4* muestra una alta especificidad y sensibilidad. Con ello en mente, el algoritmo propuesto en esta tesis, podría ser potencialmente aplicado a identificar nuevos ARN-lnc con estructuras similares a los asociados con tumores malignos.

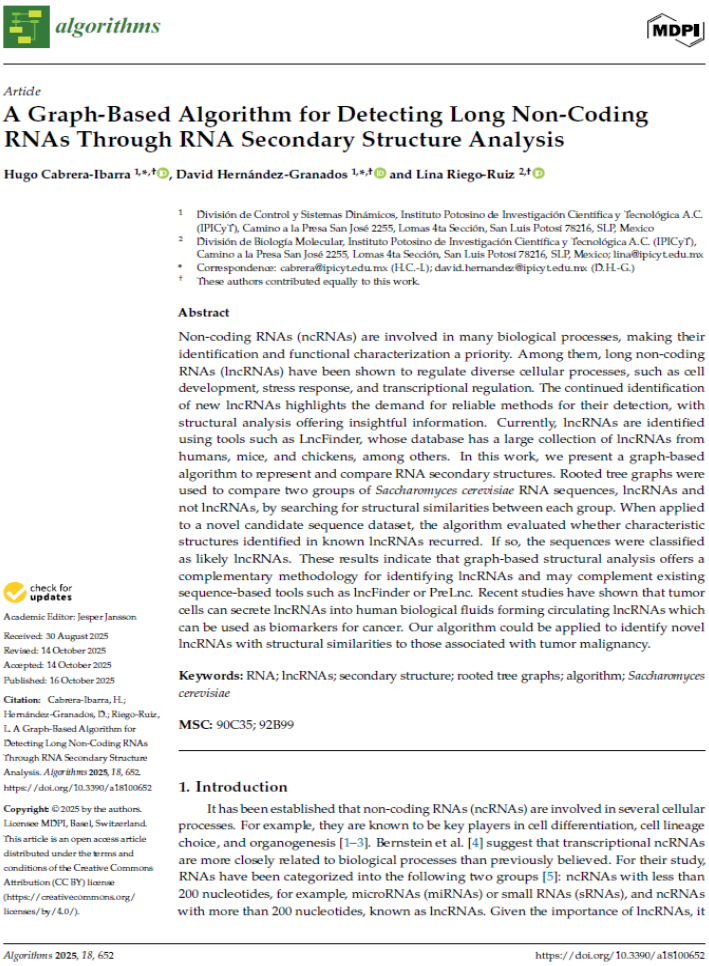
Como trabajo a futuro, se buscará mejorar las predicciones del algoritmo incorporando algunas otras características. Por ejemplo: analizando la energía libre, cuantificando los motivos estructurales, homogenizando los conjuntos para analizar, buscar si están presentes los marcos de lectura abiertos (ORFs) o inclusive analizar la preservación de las regiones exónicas. Al mismo tiempo, explorar el análisis de secuencias de ARNs provenientes de otros organismos, por ejemplo: humanos y plantas. Del mismo modo, dado que se mostró que el algoritmo es de utilidad para la clasificación de conjuntos de ARN con alguna propiedad que relaciona su estructura con su funcionalidad, se trabajará en automatizar la elección de los parámetros de control: l_1, l_2, l_3, l_4 y l_5, con el fin de mejorar tanto los tiempos de ejecución del algoritmo, así como las predicciones; para dicha tarea, se realizará un análisis estadístico más profundo con la finalidad de establecer los parámetros que permitan alcanzar el objetivo de control deseado. Así, con todo ello, ampliar la utilidad y aplicabilidad de la metodología y del algoritmo propuestos en esta tesis.

Apéndice A

Productividad

Publicaciones en revistas

• Cabrera-Ibarra, H., Hernández-Granados, D., & Riego-Ruiz, L. (2025). A Graph-Based Algorithm for Detecting Long Non-Coding RNAs Through RNA Secondary Structure Analysis. Algorithms, 18 (10), 652. https://doi.org/10.3390/a18100652.



algorithms

MDPI

Article

A Graph-Based Algorithm for Detecting Long Non-Coding RNAs Through RNA Secondary Structure Analysis

Hugo Cabrera-Ibarra [1,*,†], David Hernández-Granados [1,*,†] and Lina Riego-Ruiz [2,†]

[1] División de Control y Sistemas Dinámicos, Instituto Potosino de Investigación Científica y Tecnológica A.C. (IPICyT), Camino a la Presa San José 2255, Lomas 4ta Sección, San Luis Potosí 78216, SLP, Mexico
[2] División de Biología Molecular, Instituto Potosino de Investigación Científica y Tecnológica A.C. (IPICyT), Camino a la Presa San José 2255, Lomas 4ta Sección, San Luis Potosí 78216, SLP, Mexico; lina@ipicyt.edu.mx
* Correspondence: cabrera@ipicyt.edu.mx (H.C.-I.); david.hernandez@ipicyt.edu.mx (D.H.-G.)
† These authors contributed equally to this work.

Abstract

Non-coding RNAs (ncRNAs) are involved in many biological processes, making their identification and functional characterization a priority. Among them, long non-coding RNAs (lncRNAs) have been shown to regulate diverse cellular processes, such as cell development, stress response, and transcriptional regulation. The continued identification of new lncRNAs highlights the demand for reliable methods for their detection, with structural analysis offering insightful information. Currently, lncRNAs are identified using tools such as LncFinder, whose database has a large collection of lncRNAs from humans, mice, and chickens, among others. In this work, we present a graph-based algorithm to represent and compare RNA secondary structures. Rooted tree graphs were used to compare two groups of *Saccharomyces cerevisiae* RNA sequences, lncRNAs and not lncRNAs, by searching for structural similarities between each group. When applied to a novel candidate sequence dataset, the algorithm evaluated whether characteristic structures identified in known lncRNAs recurred. If so, the sequences were classified as likely lncRNAs. These results indicate that graph-based structural analysis offers a complementary methodology for identifying lncRNAs and may complement existing sequence-based tools such as lncFinder or PreLnc. Recent studies have shown that tumor cells can secrete lncRNAs into human biological fluids forming circulating lncRNAs which can be used as biomarkers for cancer. Our algorithm could be applied to identify novel lncRNAs with structural similarities to those associated with tumor malignancy.

Keywords: RNA; lncRNAs; secondary structure; rooted tree graphs; algorithm; *Saccharomyces cerevisiae*

MSC: 90C35; 92B99

check for updates

Academic Editor: Jesper Jansson

Received: 30 August 2025
Revised: 14 October 2025
Accepted: 14 October 2025
Published: 16 October 2025

Citation: Cabrera-Ibarra, H.; Hernández-Granados, D.; Riego-Ruiz, L. A Graph-Based Algorithm for Detecting Long Non-Coding RNAs Through RNA Secondary Structure Analysis. *Algorithms* **2025**, *18*, 652. https://doi.org/10.3390/a18100652

1. Introduction

It has been established that non-coding RNAs (ncRNAs) are involved in several cellular processes. For example, they are known to be key players in cell differentiation, cell lineage choice, and organogenesis [1–3]. Bernstein et al. [4] suggest that transcriptional ncRNAs are more closely related to biological processes than previously believed. For their study, RNAs have been categorized into the following two groups [5]: ncRNAs with less than 200 nucleotides, for example, microRNAs (miRNAs) or small RNAs (sRNAs), and ncRNAs with more than 200 nucleotides, known as lncRNAs. Given the importance of lncRNAs, it

Premios

• Por realizar una exposición excelente en el evento: *The 11th KOOK-TAPU Join Seminar on Knots and Related Topics and The 13th Graduate Student Workshop on Mathematics* realizado en Osaka, Japón, 2019.

Participación en congresos

• LVI Congreso Nacional de la SMM. Video plática por contribución: *Detectando características funcionales en conjuntos de ARNs*. S.L.P., México, 2023.

• LV Congreso Nacional de la SMM. Video plática por contribución: *Algoritmo para detectar características funcionales en el ARN mediante estructuras secundarias*. Guadalajara, México, 2022.

• XXXII Semana Nacional de Investigación y Docencia en Matemáticas. Video plática por contribución: *Análisis gráfico para la caracterización de ARNs largos no codificantes*. Sonora, México, 2022.

• LII Congreso Nacional de la SMM. Plática por contribución: *Usando grafos para la caracterización de lncRNAs*. Nuevo León, México, 2019.

• The 11th KOOK-TAPU Join Seminar on Knots and Related Topics and The 13th Graduate Student Workshop on Mathematics. Plática por contribución: *Using graphs to the characterization of lncRNAs*. Osaka, Japón, 2019.

Divulgación

• Seminario de la División de Control y Sistemas Dinámicos, IPICYT. Plática por contribución: *Un algoritmo basado en estructuras secundarias para la detección de características funcionales en el ARN: caso ARNs largos no codificantes*. San Luis Potosí, México, 2025.

• Seminario de la División de Control y Sistemas Dinámicos, IPICYT. Plática por contribución: *Algoritmo basado en estructuras secundarias para la detección de características funcionales en el ARN*. San Luis Potosí, México, 2024.

• Seminario de la División de Control y Sistemas Dinámicos, IPICYT. Plática por invitación: *Análisis gráfico para la caracterización de ARNs largos no codificantes*. San Luis Potosí, México, 2022.

• Participar en la actividad "*Huachicientíficos*", en el Museo Laberinto de las Ciencias y las Artes, S.L.P., México, 2018.

Apéndice B

Estructura primaria del ARN

La estructura primaria del ARN se refiere a la secuencia que describe a dicho ARN mediante la disposición de las cuatro bases nitrogenadas que lo componen: adenina (A), citosina (C), guanina (G) y uracilo (U) [28], la cual se representa mediante los datos de la secuencia contenidos en el formato FASTA.

Un primer acercamiento que se realizó para establecer una caracterización desde un enfoque teórico, fue determinar si dados dos grupos de ARNs, con ciertas propiedades cada uno, se podían diferenciar por medio del número total de cada base nitrogenada que los componen; para ello, se contó el número total de bases nitrogenadas A, C, G y U en los datos de las secuencias que conforman los grupos control $\bar{A}$ y $\bar{B}$, del Cuadro 2.1, y se analizaron los resultados.

Así, se contó la cantidad de letras: A, C, G y U de cada secuencia de ARN que conforman los grupos control y se normalizo el resultado, el Ejemplo 13 muestra como se llevo a cabo dicho conteo usando como base el ARN *ICR1* del grupo control $\bar{A}$.

Ejemplo 13. *Usando como base la secuencia del ARN ICR1, se contó el número de letras A que lo conforman, en este caso 919, mientras que la longitud de su secuencia fue de 3199 nucleótidos. Así, para normalizar el resultado y obtener el porcentaje de letras A que la conforman se calculó* N_A*:*

N_A *= (número de A's / longitud de la secuencia)*100 = (919/3199)*100 = 28.73*

Es decir, el 28.73 *% de la secuencia está conformada por adenina (A).*

Ahora, como en el Ejemplo 13, se realizó el calculó para todas las cadenas de dicho grupo analizando las cuatro bases nitrogenadas y se obtuvo su suma porcentual, los resultados se muestran en la Figura B.1 (A). De manera análoga, se realizó el análisis para las cadenas del grupo control $\bar{B}$, los resultados se muestran en la Figura B.1 (B).

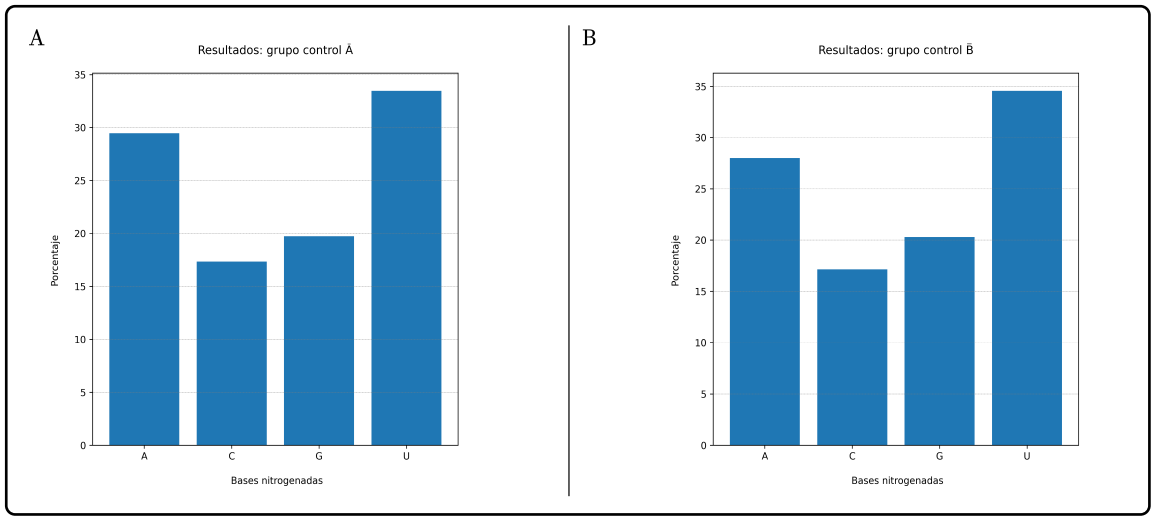


Figura B.1: Porcentaje de nucleótidos en el grupo control $\bar{A}$ (en A) y $\bar{B}$ (en B).

Como se puede observar, no existe una diferencia notoria entre el porcentaje de nucleótidos que conforman a los elementos del grupo control $\bar{A}$ y $\bar{B}$. Así, con este análisis no se puede establecer una caracterización.

Por otro lado, dado que buscar y cuantificar la existencia de cada letra no proporcionó mucha información, se buscó otra forma de abordar este problema. Para ello, ahora se analizó la búsqueda de letras, pero formando palabras, es decir, concatenando algunas de las cuatro letras y ver si estas están contenidas o no en alguna otra secuencia. Es claro, que la palabra de longitud uno es solo una letra en este caso: A, C, G o U. Notar, que a menor longitud de la palabra a buscar la probabilidad de que aparezca en otra u otras cadenas es mayor. Así, lo conveniente es buscar palabras no tan cortas para tratar de establecer una relación entre las secuencias.

Tomando como referencia los grupos control $\bar{A}$ y $\bar{B}$, del Cuadro 2.1, se buscó el número de palabras con longitud L que se comparten entre cada grupo control. Los resultados se muestran en el Cuadro B.1.

Cuadro B.1: Número de palabras encontradas.

	Número de palabras de longitud L				
Grupo control	$L \geq 10$	$L \geq 15$	$L \geq 30$	$L \geq 45$	$L \geq 100$
$\bar{A}$	200	1	0	0	0
$\bar{B}$	140	12	8	6	4

A saber, en el grupo control $\bar{A}$ se encontraron 200 palabras de longitud $L \geq 10$ que se compartían entre sus secuencias, mientras, que en el grupo control $\bar{B}$ se encontraron 140 palabras. Lo cual era de esperarse, dado que una cadena de longitud 10 es una cadena corta y por ende muy probable de que aparezca varias veces. Tratando de mejorar la búsqueda, se varió la longitud de las palabras a buscar y por ejemplo, al buscar palabras de longitud

$L \geq 15$ se obtuvo que en el grupo control $\bar{A}$ solo se encontró una palabra de esa longitud, mientras, que en el grupo control $\bar{B}$ se encontraron 12 palabras. Los resultados en el Cuadro B.1 sugieren que la probabilidad de que una palabra aparezca se reduce a medida que aumentamos un poco la longitud de la palabra a buscar. Así, no se pudieron establecer palabras que aportaran información para poder hacer una distinción entre los dos grupos analizados. Por otro lado, a lo más se compartían secuencias de longitud $L = 16$. Sin embargo, en el grupo control $\bar{B}$ hay algunas secuencias de longitud $L \geq 16$ y esto se debe a que algunas secuencias que conforman dicho grupo son muy parecidas unas con otras, por ejemplo: el ARN *SNR30* tiene solo tres nucleótidos más que el ARN *Small Nucleolar SNR30*, de 609 y 606 nucleótidos respectivamente, así, en este caso la cadena que comparten es una de 606 nucleótidos de longitud. Casos similares ocurren al comparar el ARN *SNR84* con el ARN *Small Nucleolar SNR84*, de 550 y 537 nucleótidos respectivamente, y el ARN *NMEI* con el ARN *RNASE MRP*, de 340 y 332 nucleótidos respectivamente. Mientras que al comparar el ARN *U3* con *Small Nucleolar U3* comparten subcadenas no tan cortas ya que difieren solo en pocos nucleótidos una de la otra.

Finalmente, con el análisis de la estructura primaria del ARN no se pudo establecer alguna forma de caracterizar a los elementos de cada grupo. Si bien, el ARN es una molécula simple, aún se trata de entender las formas en que realiza muchas de sus funciones. Hasta ahora, se sabe que las modificaciones postranscripcionales y su estructura secundaria son de gran importancia en su función [18, 32, 38, 39, 40, 41]. Por ello, se opto en analizar la estructura secundaria del ARN.

Apéndice C

NUPACK

El **Nucleic Acid Package** (NUPACK) es un programa que posee un conjunto de utilidades para el análisis y diseño del ácido nucleico, basándose en la estructura secundaria. Teniendo la posibilidad; de ejecutarlo a través de internet desde la dirección electrónica: http://www.nupack.org/partition/new o de solicitar el código fuente, para fines académicos y no comerciales, en la dirección electrónica: https://www.nupack.org/download/overview.

Si se ejecuta a través de internet, para generar la estructura secundaria de una secuencia de ARN se requiere ingresar a la dirección previamente mencionada y hacer clic en el menú Utilities, use como guía la Figura C.1.

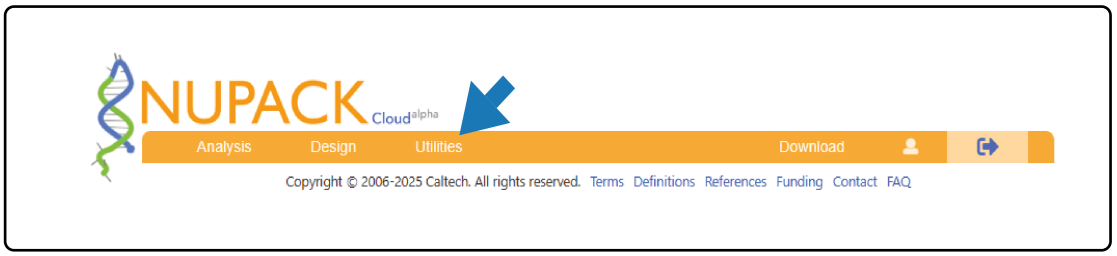


Figura C.1: Interfaz de NUPACK Web.

A manera de ejemplo se generará la estructura secundaria de la siguiente secuencia de ARN:

UUGGCACUGCCGCCAAGCGGCAGUGCCG.

Para ello, una vez seleccionado el menú Utilities, en la nueva ventana se selecciona la casilla MFE del apartado Structure (1), se ingresa la secuencia de ARN en formato FASTA en el apartado Sequence (2). Finalmente, se hace clic en el botón Update (3), cabe señalar que se dejaran fijos los demás parámetros iniciales, use de referencia la Figura C.2.

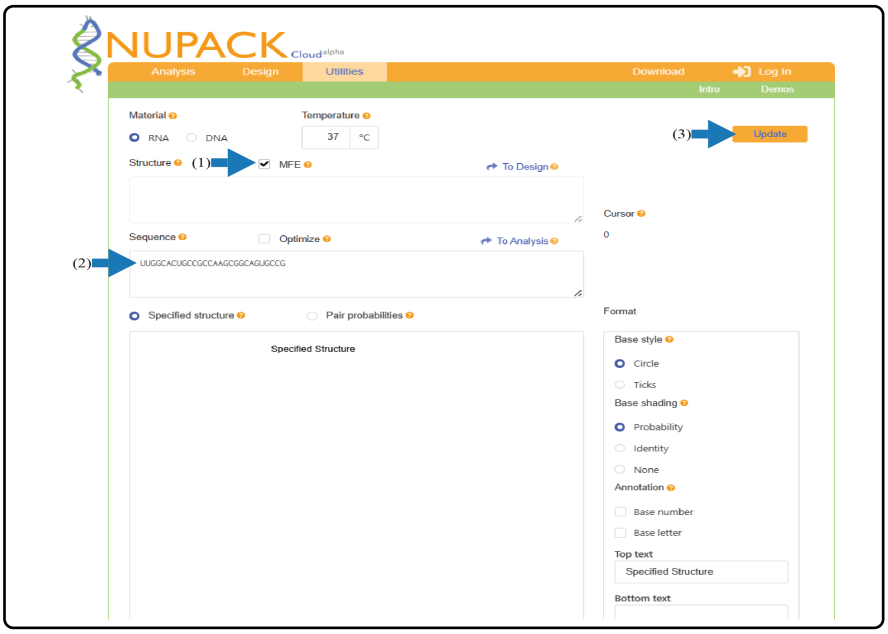


Figura C.2: Generando una estructura secundaria en NUPACK Web.

Así, al actualizarse la ventana con la información antes proporcionada, se mostrará una ventana similar a la mostrada en la Figura C.3. En la cual, se podrá observar tanto la cadena en *NPP* (4), cómo la estructura secundaria (5) correspondientes a dicha secuencia de ARN.

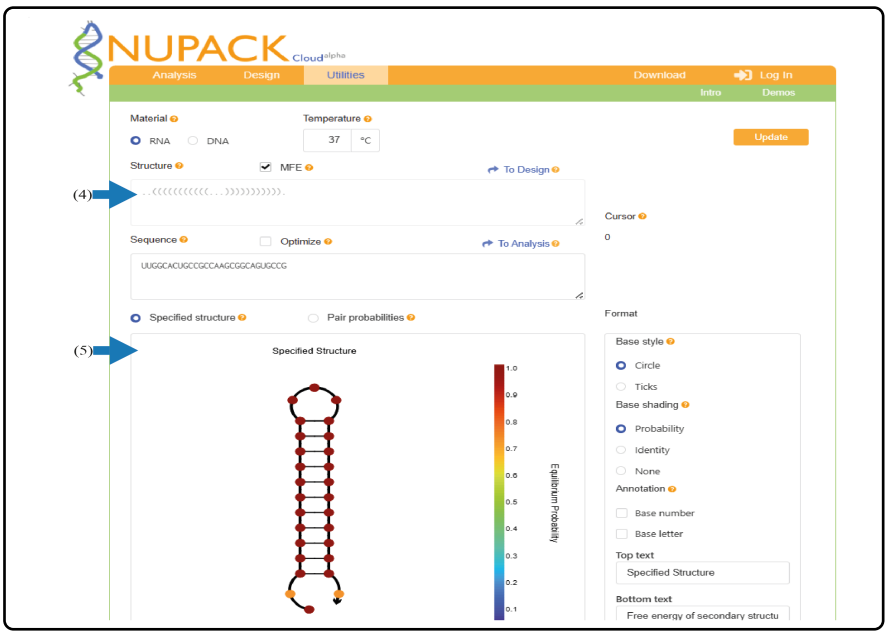


Figura C.3: Estructura secundaria mostrada por NUPACK Web.

Se pueden personalizar algunas opciones en la estructura secundaria obtenida, esto, mediante el apartado Format. Para obtener la Figura C.4 en el apartado Format (6), se elegirán las opciones: Circle (7), Identity (8) y se le cambiará el nombre a Estructura Secundaria (9).

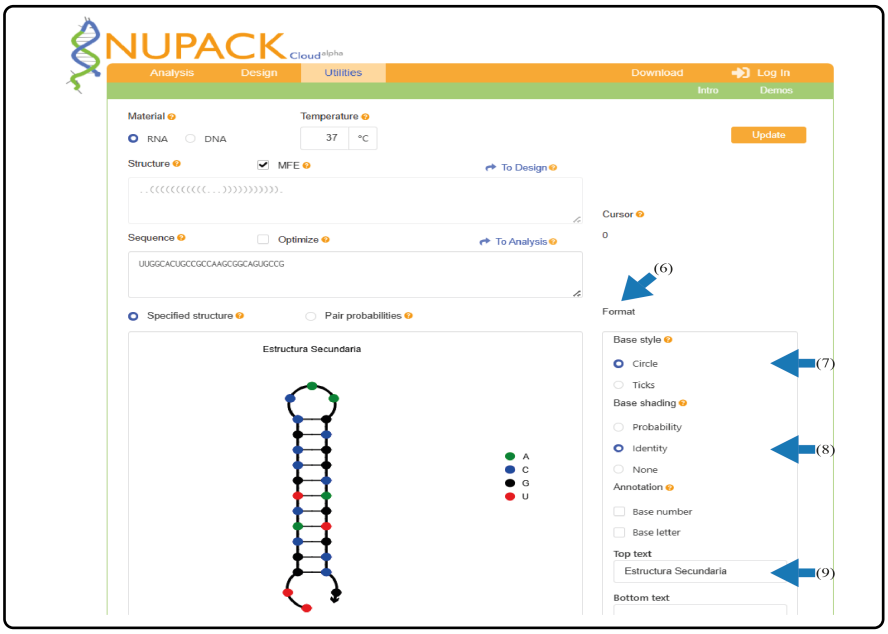


Figura C.4: Estructura secundaria mostrada por NUPACK Web usando la opción de personalización.

Así, partiendo de los datos de una secuencia de ARN se le asocio su estructura secundaria y su *NPP*. De este modo, en esta tesis, se generaron los grupos control $\bar{A}$ y $\bar{B}$, y el grupo de prueba $\bar{C}$.

Por otro lado, si solicitar el código fuente, se debe acceder a la dirección electrónica antes mencionada, llenar una serie de datos personales y aceptar los términos y condiciones. Con ello, se podrá descargar el código fuente de NUPACK, para ejecutar dicho código es necesario tener el programa Python v.3.0 instalado e importar desde la ventana de comandos NUPACK. En caso de que la importación falle o de no tener el programa de Python instalado, se puede usar una herramienta en línea, que proporcione una máquina virtual basada en Python, para con ello, poder acceder a las utilidades que ofrece NUPACK, por ejemplo: Google Collaboratory.

Apéndice D

Secuencias de ARNs de *Homo sapiens*

El Cuadro D.1 muestra parte de los datos de las secuencias de ARN analizadas, así como el número de nucleótidos que las conforman (longitud, $|L|$).

Cuadro D.1: ARNs de *Homo sapiens* que conforman los tres grupos analizados.

Grupo control $\bar{A}$		
ARN	SECUENCIA	$\|L\|$
lnc-RNF227-2:1	CCUGUCUGAAGCUAAUGCCUAUACUUGGGCUCUUAGUCUCACCCC···	2849
PCED1B-AS1:4	AAGACUAAUGACCGAAGCUUCAAGACUUUGACAUGUGUGAAAACA···	2494
LINC00200:1	GUCGCACGCUUUGCGUAGAUGAGGGAUGACGCAGGCACAGGCCUG···	2434
MIR3663HG:2	GCGGUGCAUGGGGCUAGAACUCAGGGCCCAGCAGGUCUGACAAUG···	2381
lnc-GRHL2-4:1	AAAAACCUGCUCUACAGGACAACUGACUAAUAAAUGUUGAAGCUC···	2350
lnc-ENPP1-1:1	ACAAAGCCUGUUUGGUGGUCUCUUCACACGGACGCGCAUGAAAUU···	2305
PITPNM2-AS1:6	AAUGUCUGGGCAGGGGCUGUAAAAUUGGAAUAGGUCCCCCCCCAC···	2191
lnc-MAPK6-3:9	UAAUAGAGACGGGGUUUCACCAUGUUGGCCGGGCUGUUCUCAAAG···	2073
PLUT:6	CGGCUCCCGGCUCCGGGAAAGACCUUCUGUUCUCUGCCGGCGCGG···	1992
LINC01725:44	ACCGUUGCUCAGAGUCCAGGCCGGUUAGGACCAGAGCCUACCCCG···	1991
lnc-MEIS2-2:1	GUUGGCUGGAGCUGGAGUAACAAGAUGGCGUCGUCGGCGGAGUGA···	1984
ZNF674-AS1:2	GCAAUGCGCACGCGCAGUGGCUUUGAAGGCGGGGCAACAAAGCCU···	1906
lnc-ABHD12B-2:1	GAGAUGGAGUCUCACUCUGUCACCCAGGCUGGAGUGCAGUGGCGC···	1894
SOCS2-AS1:1	AAUUUGGGAACGAGAUGCCCGGGGAUGCAACCAGCCCCGCGCUCG···	1655
lnc-GPR119-1:1	UACUCUGGCCACCACGAUCUUGGCAGCCCACACCUGAUCAGGACC···	1646
lnc-ZNF681-2:1	AUUGGAGGAGCUGCUCCCUUCCCCGUUCAGCUGUAAUUGUCUGAA···	1585
LINC00958:1	CUCUCUCUCUCUCCUGCUGCAUUGUGAAGAAACUGCUUGCUUC···	1558
MRGPRF-AS1:4	CUGCUUAACAGCUUGGAUUGAAGAGGCUUGUCUGGGGGGACUGAG···	1488
lnc-ITIH2-3:1	AACUCCCGAGGAAGUUUAAAAAAAAAAAAAUAACUUCCGUCAGCA···	1414
lnc-MYO19-1:1	GGCCGCGCUGGGGACCCCGCUGGAGAAAUUUGCGCCCAGAGGGUG···	1365
lnc-OR1J2-2:1	UUGGCUACCAGGUCAGCUAGAGAAAGACCAUCAGGUAGGGGCAGG···	1351
DCTN1-AS1:5	AGCCAACAAAGCCACAGCCCUAUCGGCUGUACUGGGCUCUGCCUU···	1273
lnc-HOMER2-2:2	GGUGAGAGCAGGCGGUGAAGAGUAGACACGACUGAUGACUUCAGG···	1261
LINC00922:3	CACCAGACCACAAACGUGGAUGGAGUUUCACUCUUAUGGCCCAGG···	1243
MALINC1:76	CAGGCUCUGGCGGGGAUAGAACCCUGAAGGGCCUGCGCCAGCUCC···	1044
lnc-NAA50-3:1	UCUGCAAUAUGCAGACACAUAAAUUAUUAUUUUAAACAUAUACUU···	1038
PIK3CD-AS2:1	UCUCUCCCUCCUUCCUGCCGUCCCCCGCCUGAGGGAGGGGAGCGG···	983
lnc-ZFHX3-27:11	GUAGAUCAUCAUCAUAACAGCUCCCAGUGAAUCAAUGCCUCCCUG···	966
IFNG-AS1:4	GGUAGCUUUUCUGACUCUUAAAGAGAUCUCAAGUAUACCUUCAGA···	902
lnc-CRYBA4-52:28	GGCGGGGGCUGCGGAGUGCUCCCUCCAAGACCGGUUGCACCCCGC···	894

lnc-KCNC2-2:3	CCCACUCGGCAGAGACCCCAGAAGGAGCCGGGGGCAGCAUCAGAG···	838
lnc-ZNF644-1:4	ACUGCAGGCUGGGGCUCCGCUCCCCGGCGGGAGCCCGCGCGUGGU···	808
LINC02327:27	CCUCAGCUUCCGGAGCAGCUGGGACUGCUGGCGCCCGCCACUACG···	777
PRR7-AS1:4	GCACGUGCCUCGGCCGUGAGGAGGAGGAGGGAUGCAAGCUUCGCU···	741
LINC02413:13	UUAUUUUGCCCUGAAAGCAGAUGAUGGUGUUAAAUAAAGCAACAA···	723
LIFR-AS1:6	CACCCAAUAGAAGGGAAAAUGGCAGAGGGAAAGUCCUAGUGGAAA···	685
lnc-CRYBA4-52:14	AUUCUUCUGGCAAACAUAUGUGGAAGGCCUCCGGGGUAGCACACC···	593
lnc-HMGB2-3:3	CGCGGGGUCAGGGAAGACCAGAGGACCACUAGCCCCAGGAAGCUU···	593
HDAC2-AS2:83	ACUGUCGCGAAGCUCCCGCCCAAGCCGACCACGUGCCGCGAGCCC···	590
ZFHX4-AS1:7	CUCAUCUUCAAUUACAAUGUGGUUCCUACAAAGUAAAGGGCUGUU···	590
STXBP5-AS1:8	CAUUCUACUUUCCGUCUCUAUGAAUUUGACUACUCGAAGUACCUC···	566
lnc-PABPC4L-5:1	UGUGAGUGUGAGAGCGAGAACAGCCCCGCAGGUCCCGAGGCGCUG···	564
lnc-CPM-2:5	GUUCCUGGUUGCGUUUAGCCCAGACGCCCUGCGGCCGCUUUGCCU···	550
lnc-FGF3-2:3	AGGUAAAAAAGAAAGAGGAAUCAAAGAAAACAAAGGAAGGGCAGA···	548
FRG1-DT:27	UAACAAAGGAGACACACAGGGGCUCCUCACGUUGGCUUUCCAGGG···	517
lnc-CIT-1:4	ACGAGGAGUGCAGAGGAGGUGCUGAUGAUUGCGGGGAGAGCUCUG···	495
LINC02372:15	CAAGACAGCACCGGGUUUCACACUGGCAAGGCAGGAGGCUGUGUG···	482
SNAP25-AS1:8	AAACUCAACAGCCUAAAUAUCUCUGAAAACUGCAUCGCAAAAUGG···	453
CACNA1G-AS1:3	CGCCGCGGGUGUGCUGGGCGCCGGUGGAGGGGGGAACAGCGAGGA···	368
LINC02447:27	CAACUCUUGAUUCUAUCACAAAACACUGCACCCGAGGCAGCCAGG···	292

Grupo control $\bar{B}$		
ARN	SECUENCIA	$\|L\|$
CCDS45516	AUGGCGGCCCGACAGGCCGUGGGCAGCGGGGCUCAGGAGACAUGC···	3285
CCDS45314	AUGCCGGGGGCCGGAGCCCGGGCGGAGGAGGGCGGCGGCGGCGGC···	2892
CCDS45012	AUGGCCACCAGGGUCCGGACAGCUUCUAUUUGGGUCCCACCUCUC···	2844
CCDS45616	AUGAAGAAGCAGUUCAAUCGCAUGCGCCAGCUGGCCAACCAGACG···	2457
CCDS45715	AUGGCUGAGCCCCGCCAGGAGUUCGAAGUGAUGGAAGAUCACGCU···	2331
CCDS45111	AUGGCUACCACGGCCGAGCUCUUCGAGGAGCCUUUUGUGGCAGAU···	2127
CCDS45419	AUGGAGAUAAUCAGGAGCAAUUUUAAGAGUAAUCUUCACAAAGUG···	1920
CCDS45412	AUGGAUCCGGGCAGUGGCGGCGGCGGCGGCGGCGGCGGCGGC···	1914
CCDS45313	AUGCAGGUGAGCAAGAGGAUGCUGGCGGGGGGCGUGAGGAGCAUG···	1863
CCDS45812	AUGGCCGAGAGCAUCAUAAUUCGUGUCCAGUCCCCGGAUGGAGUG···	1827
CCDS45211	AUGGCAGAAGAAACUGGACAGAGUAAAUUAGCUGCAGCCAAGAAA···	1812
CCDS45615	AUGGCGCUGCGGCGCCUCCUGCUGCUGCUGCUGCUCUCGCUGGAG···	1656
CCDS45810	AUGCCGACGAACGGCCUGCACCAGGUGCUGAAGAUCCAGUUUGGC···	1551
CCDS45414	AUGGAACUUUCAGAACCUAUUGUAGAAAAUGGAGAGACAGAAAUG···	1500
CCDS13539	AUGUGCGUCAUCUGCUUCGUGAAGGCGCUGGUGCGCGUGUUCAAG···	1410
CCDS45510	AUGCAGGAAGCGCCAGCUGCGCUGCCCACGGAGCCAGGCCCCAGC···	1389
CCDS45315	AUGGAGGACAGCCUAAAGCAGCUCAGCCUGGGGAGAGAUCCUGAG···	1356
CCDS45712	AUGGGAGGGCACCCGCAGCUCCGUCUCGUCAAGGCCCUUCUCCUU···	1335
CCDS45517	AUGACAACGUCAACCCUCCAGAAAGCCAUUGAUCUGGUGACGAAA···	1314
CCDS6772	AUGUGGGCUCCACCAGCAGCAAUCAUGGGGGAUGGGCCCACCAAG···	1308
CCDS45417	AUGCGCCGCGUCCUUCGGCUGCUCCUCGGUUGCUUCCUCACCGAG···	1275
CCDS12207	AUGGAUGCAAGUCGCCCCAAGUCCUCGGAAUCCCAGUCCUCCCUG···	1251
CCDS45312	AUGAUGCCCCAGCUGCAGUUCAAAGAUGCCUUUUGGUGCAGGGAC···	1251
CCDS6771	AUGGCGACAAGGAACAGCCCCAUGCCCCUGGGCACGGCUCAGGGU···	1248
CCDS45013	AUGAAUGAAAGUCCACAGACAAACGAAUUUAAAGGAACAACCGAG···	1236
CCDS45217	AUGGCGCAAAGGUACGAUGAGCUGCCCCAUUACGGCGGGAUGGAC···	1206
CCDS45316	AUGGUGAAGGAGACCCAGUACUAUGACAUCCUGGGCGUGAAGCCC···	1194
CCDS45219	AUGGACGGAGUAGGGGUUCCCGCUUCCAUGUACGGAGACCCUCAC···	1167
CCDS2706	AUGGCCGCCACUGCCUCUCCGCAGCCACUCGCCACUGAGGAUGCC···	1155
CCDS45	AUGUUUAAUCCGCACGCUUUAGACUCCCCGGCUGUGAUUUUUGAC···	1134

lnc-FAM49B-1:1	GUGUGGCUCAUGCCUGUAGUUCCAGCACUUUGGGAGGCCAAGG···	2184
lnc-ANP32A-3	AACUGGCCUGUGUAAGCGGAUGGGCUCGGCCUCAAGGACAGGGUG···	1778
lnc-JAM3-3	CUCGGCAGCCGGCCUUAGCCUGGAGGAGCAACCCUGGUCCAGAGA···	1673
LINC02447:25	CUCGGAGCCCGGCGCAGUGGAAAGCGGGCCAAGAAGCAGAGGUCC···	1500
LINC00158:1	GGAUUAAUCUAAGCUUCUGCUGUGUAUGGACCUGUGGCUACAUUA···	1380
lnc-INTS14-1:2	CACAGUUUAUUGAAAUUUAAUAAAAAUUAUUUCCAAAGAAUGAGA···	1374
LINC01366	AGGAGUUGCUGAGAAGGAGCUGUGGUUCUGACCUUCAGGGGAAUU···	725
lnc-NXPH1-2	AGACUGCUGUGCUAGCAAUCAGCGAGACUCCGUGGGCGUAGGACC···	557
lnc-DHX37-9:4	ACCCAUGCUGGGCUAAGACAAGGGGCAGAGGCAGAGAGAGAGAGA···	419
LINC01725	CCGGCCUGUGCGCAGUUCCCGUGCCCUAGGAGUUGGCGGCGGGGC···	400
CCDS9804	AUGGCAGUGGAUGGGACCCUCGUGUACAUCAGAGUCACUCUUCUG···	2169
CCDS45914	AUGGAUGAGUUCCACCCGUUCAUCGAGGCCCUGCUGCCUCACGUC···	1500
CCDS45916	AUGGACCAGGACUAUGAGCGGCGCCUGCUUCGCCAGAUCGUCAUC···	1491
CCDS7586	AUGGGCGCGGGGGUGCUCGUCCUGGGCGCCUCCGAGCCCGGUAAC···	1434
CCDS7569	AUGUUCCGCCAGGAGCAGCCGUUGGCCGAGGGCAGCUUUGCGCCC···	1398
CCDS56129	AUGGACCACCAGGACCCCUACUCCGUGCAGGCCACAGCGGCCAUA···	1353
CCDS4292	AUGGGGCAACCCGGGAACGGCAGCGCCUUCUUGCUGGCACCCAAU···	1242
CCDS13826	AUGCCCAUCAUGGGCUCCUCGGUGUACAUCACGGUGGAGCUGGCC···	1239
CCDS82854	AUGAAGAAAAUGAAUCACAAGUCAACUGACAGUCCAAAGGCUCCA···	1185
CCDS3137	AUGAUUCUCAACUCUUCUACUGAAGAUGGUAUUAAAAGAAUCCAA···	1080

Grupo de prueba $\bar{C}$		
ARN	SECUENCIA	$\|L\|$
lnc-FAM49B-1:1	GUGUGGCUCAUGCCUGUAGUUCCAGCACUUUGGGAGGCCAAGGGA···	2184
lnc-ANP32A-3	AACUGGCCUGUGUAAGCGGAUGGGCUCGGCCUCAAGGACAGGGUG···	1778
lnc-JAM3-3	CUCGGCAGCCGGCCUUAGCCUGGAGGAGCAACCCUGGUCCAGAGA···	1673
LINC02447:25	CUCGGAGCCCGGCGCAGUGGAAAGCGGGCCAAGAAGCAGAGGUCC···	1500
LINC00158:1	GGAUUAAUCUAAGCUUCUGCUGUGUAUGGACCUGUGGCUACAUUA···	1380
lnc-INTS14-1:2	CACAGUUUAUUGAAAUUUAAUAAAAAUUAUUUCCAAAGAAUGAGA···	1374
LINC01366	AGGAGUUGCUGAGAAGGAGCUGUGGUUCUGACCUUCAGGGGAAUU···	725
lnc-NXPH1-2	AGACUGCUGUGCUAGCAAUCAGCGAGACUCCGUGGGCGUAGGACC···	557
lnc-DHX37-9:4	ACCCAUGCUGGGCUAAGACAAGGGGCAGAGGCAGAGAGAGAGAGA···	419
LINC01725	CCGGCCUGUGCGCAGUUCCCGUGCCCUAGGAGUUGGCGGCGGGGC···	400
CCDS9804	AUGGCAGUGGAUGGGACCCUCGUGUACAUCAGAGUCACUCUUCUG···	2169
CCDS45914	AUGGAUGAGUUCCACCCGUUCAUCGAGGCCCUGCUGCCUCACGUC···	1500
CCDS45916	AUGGACCAGGACUAUGAGCGGCGCCUGCUUCGCCAGAUCGUCAUC···	1491
CCDS7586	AUGGGCGCGGGGGUGCUCGUCCUGGGCGCCUCCGAGCCCGGUAAC···	1434
CCDS7569	AUGUUCCGCCAGGAGCAGCCGUUGGCCGAGGGCAGCUUUGCGCCC···	1398
CCDS56129	AUGGACCACCAGGACCCCUACUCCGUGCAGGCCACAGCGGCCAUA···	1353
CCDS4292	AUGGGGCAACCCGGGAACGGCAGCGCCUUCUUGCUGGCACCCAAU···	1242
CCDS13826	AUGCCCAUCAUGGGCUCCUCGGUGUACAUCACGGUGGAGCUGGCC···	1239
CCDS82854	AUGAAGAAAAUGAAUCACAAGUCAACUGACAGUCCAAAGGCUCCA···	1185
CCDS3137	AUGAUUCUCAACUCUUCUACUGAAGAUGGUAUUAAAAGAAUCCAA···	1080

Apéndice E

Pseudocódigo

En esta sección, se muestra el pseudocódigo del algoritmo para que pueda ser implementado en diferentes lenguajes de programación. Se utilizan como datos de entrada los conjuntos $\hat{A}$, $\hat{B}$ y $\hat{C}$, de ARNs-lnc, ARNs que no pertenecen a los ARNs-lnc y un conjunto de ARN candidatos a ser ARNs-lnc, respectivamente.

▷ **Pseudocódigo:**

1: Usar un programa de plegamiento para obtener las cadenas en *NPP* de los ARNs a analizar descritos por los datos de su secuencia, en el formato FASTA.
2: Simplificar, usando las reglas de Gan *et al.* [4], las cadenas en *NPP* a cadenas en *NPPS*. Obteniendo con ello, los conjuntos A, B y C, respectivamente.
3: Dados l_1 y l_2 generar los conjuntos S_A (y S_B) de cadenas repetidas en A (y en B) de longitud l que satisfagan $l_1 \leq l \leq l_2$.
4: Usar S_A (y S_B) para calcular la matriz de cadenas repetidas presentes en A, denotada como M_{AA} (similarmente, calcular M_{AB}, M_{BB} y M_{BA}).
5: Establecer un discriminante adecuado l_3, en esta tesis se toma $l_3 = 2$, y sumar los elementos de cada columna de la matriz M_{AA} para obtener C_{AA} (similarmente, calcular C_{AB}, C_{BB} y C_{BA}). Evaluar $C_{AA} - C_{AB} \geq l_3$ para determinar el conjunto de cadenas relevantes $S_A^{'}$ de A (de igual forma, evaluar $C_{BB} - C_{BA} \geq l_3$, $S_B^{'}$ de B).
6: Como en el paso (4), pero ahora con los conjuntos $S_A^{'}$ ($S_B^{'}$) y $D = A + B + C$ calcular la matriz de cadenas relevantes presentes en A, denotada $M_{AA}^{'}$ (similarmente, calcular $M_{BB}^{'}$, $M_{AD}^{'}$ y $M_{BD}^{'}$). Luego, sumar los elementos de cada fila de la matriz R_{AA} (similarmente, calcular R_{AD}, R_{BB} y R_{BD}), donde la i-ésima entrada de R_{AA} es el número de apariciones de elementos de $S_A^{'}$ presentes en el i-ésimo elemento de A. Además, calcular el primer cuartil $Q1A$ de R_{AA} y el primer cuartil $Q1B$ de R_{BB}.
7: Determinar si la i-ésima entrada de R_{ADi} en R_{AD} (y R_{BDi} en R_{BD}) satisfacen ambas restricciones: $R_{ADi} > 0.5 * Q1_A$ y $R_{BDi} < 1.5 * Q1_B$. De ser así, dicho elemento en D es detectado como un posible ARN-lnc.
8: Sea el número N_A (y N_B) de elementos de A (y B) detectados como ARNs-lnc. Cuando N_A y N_B satisfagan $N_A \geq 7$ y $N_B \leq 3$. Entonces, los elementos de C detectados serán más probables de ser ARNs-lnc, V_C.

Bibliografía

[1] Schlick, T. (2018). Adventures with RNA graphs. *Methods, 143*, 16-633. https://doi.org/10.1016/j.ymeth.2018.03.009.

[2] Joyce, G. F. (1994). In vitro evolution of nucleic acids. *Current opinion in structural biology, 4*, 331-336. https://doi.org/10.1016/s0959-440x(94)90100-7.

[3] Liu, P., Lusk, J., Jonoska, N., & Vázquez, M. (2024). Tree polynomials identify a link between co-transcriptional R-loops and nascent RNA folding. *Plos. Comput. Biol., 20*, 1-24. https://doi.org/10.1371/journal.pcbi.1012669.

[4] Gan, H. H., Pasquali, S., & Schlick, T. (2003). Exploring the repertoire of RNA secondary motifs using graph theory: implications for RNA design. *Nucleic Acids Research, 31 (11)*, 2926-2943. https://doi.org/10.1093/nar/gkg365.

[5] Mattick, J. S. (2018). The State of Long Non-Coding RNA Biology. *Non-Coding RNA, 4 (3)*, 17. https://doi.org/10.3390/ncrna4030017.

[6] Zamore, P. D., & Haley, B. (2005). Ribo-gnome: the big world of small RNAs. *Science, 309 (5740)*, 1519-1524. https://doi.org/10.1126/science.1111444.

[7] Huang, T., Alvarez, A., Hu, B., & Cheng, S. Y. (2013). Noncoding RNAs in cancer and cancer stem cells. *Chinese journal of cancer, 32 (11)*, 582-593. https://doi.org/10.5732/cjc.013.10170.

[8] Cheetham, S. W., Gruhl, F., Mattick, J. S., & Dinger, M. E. (2013). Long noncoding RNAs and the genetics of cancer. *British journal of cancer, 108 (12)*, 2419-2425. https://doi.org/10.1038/bjc.2013.233.

[9] Baylin, S., & Jones, P. (2011). A decade of exploring the cancer epigenome-biological and translational implications. *Nat. Re.v Cancer, 11*, 726-734. https://doi.org/10.1038/nrc3130.

[10] Bugnon, L. A., Edera, A. A., Prochetto, S., Gerard, M., Raad, J., Fenoy, E., Rubiolo, M., Chorostecki U., Gabaldón, T., Ariel, F., DiPersia, L. E., Milone, D. H., & Stegmayer, G. (2022). Secondary structure prediction of long noncodingRNA:review and experimental comparison of existing approaches. *Briefings in Bioinformatics, 23 (4)*, 1-14. https://doi.org/10.1093/bib/bbac205.

[11] Mamuye, A., Rucco, M., Tesei, L. & Merelli, E. (2016). Persistent Homology Analysis of RNA. *Computational and Mathematical Biophysics, 4 (1)*, 14-25. https://doi.org/10.1515/mlbmb-2016-0002.

[12] Siyu, H., Yanchun, L., Qin, M., Yangyi, X., Yu, Z., Wei, D., Cankun, W., & Ying, L. (2019). LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information and physicochemical property. *Brief. Bioinform 20 (6)*, 2009-2027. https://doi.org/10.1093/bib/bby065.
Site: http://bmbl.sdstate.edu/lncfinder

[13] Cao, L., Wang, Y., Bi, C., Ye, Q., Yin, T., & Ye, N. (2020). PreLnc: An Accurate Tool for Predicting lncRNAs Based on Multiple Features. *Genes, 11 (9)*, 981. https://doi.org/10.3390/genes11090981.
Site: https://github.com/LeiCao97/PreLnc

[14] Eddy, S. R. (2001). Non-coding RNA genes and the modern RNA world. *Nat. Rev. Genet. 2*, 919-929. https://doi.org/10.1038/35103511.

[15] Waters, L. S., & Storz, G. (2009). Regulatory RNAs in bacteria. *Cell, 136 (4)*, 615-628. https://doi.org/10.1016/j.cell.2009.01.043.

[16] Fernandes, J. C. R., Acuña, S. M., Aoki, J. I., Floeter-Winter, L. M., & Muxel, S. M. (2019). Long Non-Coding RNAs in the Regulation of Gene Expression: Physiology and Disease. *Non-coding RNA, 5 (1)*, 17. https://doi.org/10.3390/ncrna5010017.

[17] Bernstein, B. E., Birney, E., Dunham, I., Green, E. D., Gunter, C., & Snyder, M. (2012). An integrated encyclopedia of DNA elements in the human genome. *Nature, 489 (7414)*, 57-74. https://doi.org/10.1038/nature11247.

[18] Zampetaki, A., Albrecht, A., & Steinhofel, K. (2018). Long Non-coding RNA Structure and Function: Is There a Link?. *Front. Physiol., 9*, 1201. doi:10.3389/fphys.2018.01201.

[19] de Andres-Pablo, A., Morillon, A. & Wery, M. (2017). LncRNAs, lost in translation or licence to regulate?. *Curr. Genet. 63*, 29-33. https://doi.org/10.1007/s00294-016-0615-1.

[20] Fang, Y., & Fullwood, M. J. (2016). Roles, Functions, and Mechanisms of Long Non-coding RNAs in Cancer. *Genomics, proteomics and bioinformatics, 14 (1)*, 42-54. https://doi.org/10.1016/j.gpb.2015.09.006.

[21] Till, P., Mach, R. L. & Mach-Aigner, A. R. (2018). A current view on long noncoding RNAs in yeast and filamentous fungi. *Appl. Microbiol. Biotechnol., 102*, 7319-7331. https://doi.org/10.1007/s00253-018-9187-y.

[22] Yamashita, A., Shichino, Y., & Yamamoto, M. (2016). The long non-coding RNA world in yeasts. *Biochimica et biophysica acta, 1859 (1)*, 147-154. https://doi.org/10.1016/j.bbagrm.2015.08.003.

[23] Botstein, D., & Fink, G. R. (1998). Yeast: an experimental organism for modern biology. *Science, 240*, 1439-1443. https://doi.org/10.1126/science.3287619.

[24] Smith, M. G., & Snyder, M. (2006), Yeast as a Model for Human Disease. *Current Protocols in Human Genetics, 48*: 15.6.1-15.6.8. https://doi.org/10.1002/0471142905.hg1506s48.

[25] Nielsen, J. (2012). Production of biopharmaceutical proteins by yeast: Advances through metabolic engineering. *Bioengineered, 4 (4)*, 207-211. https://doi.org/10.4161/bioe.22856.

[26] Sadava, D. E., Hillis, D., Heller, H. C., & Berenbaum, M. R. (2014). Life: The Science of Biology. *Macmillan (10 ed.)*.

[27] Verman, P. S., & Agarwal, V. K. (2005). Cell biology, genetics, molecular biology, evolution and ecology. *S. Chand and Company LTD*.

[28] Jiménez, L. F., & Merchant, H. (2003). Biología celular y molecular. *Pearson Educación*.

[29] Grabow, W. W., & Andrews, G. E. (2019). On the nature and origin of biological information: The curious case of RNA. *BioSystems, 185*. https://doi.org/10.1016/j.biosystems.2019.104031.

[30] Jcfidy-File: Difference DNA and RNA-EN.svg, consultado en octubre de 2025. https://commons.wikimedia.org/w/index.php?curid=20525524.

[31] Hüttenhofer, A., Schattner, P., & Polacek, N. (2005). Non-coding RNAs: hope or hype?. *Trends in genetics: TIG, 21 (5)*, 289-297. https://doi.org/10.1016/j.tig.2005.03.007.

[32] Mei-zhen, L., Hua-mei, X., Kang, H., Fei, L. (2019). Progress and prospects of noncoding RNAs in insects. *Journal of Integrative Agriculture, 18 (4)*, 729-747.

[33] Fang, Y., & Fullwood, M. J. (2016). Roles, functions, and mechanisms of long noncoding RNAs in cancer. Genomics, *Proteomics and Bioinformatics, 14 (1)*, 42-54, https://doi.org/10.1016/j.gpb.2015.09.006.

[34] Niederer, R. O., Hass, E. P., & Zappulla, D. C. (2017). Long Noncoding RNAs in the Yeast S. cerevisiae . *Long Non Coding RNA Biology. Advances in Experimental Medicine and Biology, 1008*, 119-132. Springer, Singapore. https://doi.org/10.1007/978-981-10-5203-3_4.

[35] van Bakel, H., Nislow, C., Blencowe, B. J., & Hughes, T. R. (2010). Most "Dark Matter" Transcripts Are Associated With Known Genes. *PLOS Biology, 8 (5):e1000371*. https://doi.org/10.1371/journal.pbio.1000371.

[36] Chowdhary, A., Satagopam, V., & Schneider, R. (2021). Long Non-coding RNAs: Mechanisms, Experimental, and Computational Approaches in Identification, Characterization, and Their Biomarker Potential in Cancer. *Front. Genet., 12:649619*. https://doi.org/10.3389/fgene.2021.649619.

[37] Ulitsky, I. (2016). Evolution to the rescue: using comparative genomics to understand long non-coding RNAs. *Nat. Rev. Genet., 17*, 601-614. https://doi.org/10.1038/nrg.2016.85.

[38] Sanbonmatsu, K. (2022). Getting to the bottom of lncRNA mechanism: structure-function relationships. *Mammalian genome, 33*, 343-353. https://doi.org/10.1007/s00335-021-09924-x.

[39] Nadhan, R., Isidoro, C., Song, Y. S., & Dhanasekaran, D. N. (2022). Signaling by LncRNAs: Structure, Cellular Homeostasis, and Disease Pathology. *Cells, 11 (16)*, 2517. https://doi.org/10.3390/cells11162517.

[40] Mattick, J. S., Amaral, P. P., Carninci, P. & *et al*. (2023). Long non-coding RNAs: definitions, functions, challenges and recommendations. *Nat. Rev. Mol. Cell. Biol., 24*, 430-447. https://doi.org/10.1038/s41580-022-00566-8.

[41] Ballarino, M., Gerardo, Pepe, Helmer-Citterich, M., & Palma, A. (2023). Exploring the landscape of tools and resources for the analysis of long non-coding RNAs. *Computational and Structural Biotechnology Journal, 21*, 4706-4716. https://doi.org/10.1016/j.csbj.2023.09.041.

[42] Mount, D. W. (2004). Bioinformatics: Sequence and Genome Analysis. *Spring Harbor Press, 2^a ed*.

[43] Tanaka, T., & Kondo, A. (2015). Cell-surface display of enzymes by the yeast Saccharomyces cerevisiae for synthetic biology. *FEMS Yeast Res., 15 (1)*, 1-9. https://doi.org/10.1111/1567-1364.12212.

[44] Nielsen, J., & Keasling, J. D. (2016). Engineering cellular metabolism. *Cell, 164 (6)*, 1185-1197. https://doi.org/10.1016/j.cell.2016.02.004.

[45] Zhao, X. Q., & Bai, F. W. (2009). Mechanisms of yeast stress tolerance and its manipulation for efficient fuel ethanol production. *Journal of biotechnology, 144 (1)*, 23-30. https://doi.org/10.1016/j.jbiotec.2009.05.001.

[46] Cherry, J. M., Hong, E. L., Amundsen, C., Balakrishnan, R., Binkley, G., Chan, E. T., Christie, K. R., Costanzo, M. C., Dwight, S. S., Engel, S. R., Fisk, D. G., Hirschman, J. E., Hitz, B. C., Karra, K., Krieger, C. J., Miyasato, S. R., Nash, R. S., Park, J., Skrzypek, M. S., Simison, M., & Wong, E. D. (2012). Saccharomyces Genome Database: the genomics resource of budding yeast. *Nucleic acids research, 40 (Database issue)*, D700-D705. https://doi.org/10.1093/nar/gkr1029.
Site: http://www.yeastgenome.org

[47] Novaičić, A., Vučenović, I., Primig, M., & Stuparević, I. (2020). Non-coding RNAs as cell wall regulators in Saccharomyces cerevisiae. *Critical Reviews in Microbiology, 46 (1)*, 15-25. https://doi.org/10.1080/1040841X.2020.1715340.

[48] Balarezo-Cisneros, L. N., Parker, S., Fraczek, M. G., Timouma, S., Wang, P., & *et al*. (2021). Functional and transcriptional profiling of non-coding RNAs in yeast reveal context-dependent phenotypes and in trans effects on the protein regulatory network. *PLOS Genetics 17 (1):e1008761*. https://doi.org/10.1371/journal.pgen.1008761.

[49] Lee, J. C., & Gutell, R. R. (2004). Diversity of base-pair conformations and their occurrence in rRNA structure and RNA structural motifs. *J. Mol. Biol., 344 (5)*, 1225-1249. https://doi.org/10.1016/j.jmb.2004.09.072.

[50] Brown, T. A. (2008). Genomas. *Médica Panamericana,* 3^a *ed*.

[51] Zadeh, J. N., Steenberg, C. D., Bois, J. S., Wolfe, B. R., Pierce, M. B., Khan, A. R., Dirks, R. M., & Pierce, N. A. (2011). NUPACK: Analysis and design of nucleic acid systems. *Journal of computational chemistry, 32 (1)*, 170-173. https://doi.org/10.1002/jcc.21596.
Site: https://www.nupack.org/

[52] Gruber, A. R., Lorenz, R., Bernhart, S. H., Neuböck, R., & Hofacker, I. L. (2008) The Vienna RNA Websuite, *Nucl. Acids Res., 36 (2)*, 70-74. https://doi.org/10.1093/nar/gkn188.
Site: http://rna.tbi.univie.ac.at/cgi-bin/RNAWebSuite/RNAfold.cgi

[53] Zuker, M. (2003). Mfold web server for nucleic acid folding and hybridization prediction, *Nucl. Acids Res., 31 (13)*, 3406-3415. https://doi.org/10.1093/nar/gkg595.
Site: https://www.unafold.org/mfold/applications/rna-folding-form.php

[54] Willmott, D., Murrugarra, D., & Ye, Q. (2020). Improving RNA secondary structure prediction via state inference with deep recurrent neural networks, *Computational and Mathematical Biophysics, 8 (1)*, 36-50. https://doi.org/10.1515/cmb-2020-0002.
Site: https://github.com/dwillmott/rna-state-inf

[55] Sato, K., Akiyama, M. & Sakakibara, Y. (2021). RNA secondary structure prediction using deep learning with thermodynamic integration. *Nat. Commun. 12, 941*, 1-10. https://doi.org/10.1038/s41467-021-21194-4.
Site: https://github.com/mxfold/mxfold2

[56] Dirks, R. M., Bois, J. S., Schaeffer, J. M., Winfree, E., & Pierce, N. A. (2007). Thermodynamic analysis of interacting nucleic acid strands. *SIAM Rev., 49 (1)*, 65-88. https://doi.org/10.1137/060651100.

[57] Achar, A., & Sætrom, P. (2015). RNA motif discovery: a computational overview. *Biol. Direct. 10 (61)*. https://doi.org/10.1186/s13062-015-0090-5.

[58] Zhu, Q., Petingi, L., & Schlick, T. (2022). RNA-As-Graphs Motif Atlas?Dual Graph Library of RNA Modules and Viral Frameshifting-Element Applications. *International Journal of Molecular Sciences, 23 (16)*, 9249. https://doi.org/10.3390/ijms23169249.

[59] Diestel, R. (2017). Graph Theory. *Springer Hamburg, 5th edition*, pp. 1-17.

[60] Staple, D. W., & Butcher, S. E. (2005). Pseudoknots: RNA Structures with Diverse Functions. *PLoS Biol., 3 (6):e213*. https://doi.org/10.1371/journal.pbio.0030213.

[61] Xu, Z., Wei, W., Gagneur, J., Perocchi, F., Clauder-Münster, S., Camblong, J., Guffanti, E., Stutz, F., Huber, W., & Steinmetz, L. M. (2009). Bidirectional promoters generate pervasive transcription in yeast. *Nature, 457 (7232)*, 1033-1037. https://doi.org/10.1038/nature07728.

[62] van Dijk, E., Chen, C., d'Aubenton-Carafa, Y., & *et al*. (2011). XUTs are a class of Xrn1-sensitive antisense regulatory non-coding RNA in yeast. *Nature, 475*, 114-117. https://doi.org/10.1038/nature10118.

[63] Geisler, S., Lojek, L., Khalil, A. M., Baker, K. E., & Coller, J. (2012). Decapping of long noncoding RNAs regulates inducible genes. *Molecular Cell, 45 (3)*, 279-291. https://doi.org/10.1016/j.molcel.2011.11.025.

[64] Castelnuovo, M., Rahman, S., & Guffanti, E. *et al*. (2013). Bimodal expression of PHO84 is modulated by early termination of antisense transcription. *Nat. Struct. Mol. Biol. 20*, 851-858. https://doi.org/10.1038/nsmb.2598.

[65] Singh, J., Paliwal, K., Zhang, T., Singh, J., Litfin, T., & Zhou, Y. (2021). Improved RNA secondary structure and tertiary base-pairing prediction using evolutionary profile, mutational coupling and two-dimensional transfer learning, *Bioinformatics, 37 (17)*, 2589-2600. https://doi.org/10.1093/bioinformatics/btab165.
https://github.com/jaswindersingh2/SPOT-RNA2

[66] Fu, L., Cao, Y., Wu, J., Peng, Q., Nie, Q., & Xie, X. (2022). UFold: fast and accurate RNA secondary structure prediction with deep learning, *Nucleic Acids Research, 50 (3)*, 14. https://doi.org/10.1093/nar/gkab1074.
http://146.56.237.198:3838/UFold

[67] Hu, X., Bao, J., Wang, Z., & *et al*. (2016). The plasma lncRNA acting as fingerprint in nonsmall-cell lung cancer. *Tumor Biol., 37 (3)*, 3497-3504. https://doi.org/10.1007/s13277-015-4023-9.

[68] Xu, N., Chen, F., Wang, F., & *et al*. (2015). Clinical significance of high expression of circulating serum lncRNA RP11-445H22.4 in breast cancer patients: a Chinese population-based study. *Tumor Biol., 36*, 7659-7665. https://doi.org/10.1007/s13277-015-3469-0.