UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA TÈCNICA SUPERIOR D´ENGINYERIA AGRONÒMICA I DEL MEDI NATURAL

Diseño e implementación de un analizador de biosecuencias: mutaciones producidas por elongación de horquilla

TRABAJO FIN DE GRADO EN BIOTECNOLOGÍA

ALUMNO: FERNANDO NAYA CATALÀ

DIRECTOR: JOSÉ MARÍA SEMPERE LUNA

TUTOR: JOAQUÍN CAÑIZARES SALES

Curso Académico: 2015-2016

VALENCIA, 6 JULIO 2016

Datos del trabajo

Título del TFG:	Diseño e implementación de un analizador de biosecuencias: Mutaciones producidas por elongación de horquilla.
Autor:	Fernando Naya Català
Localidad y fecha:	Valencia, 6 de Julio 2016
Tutor:	Santiago Cañizares Sales
Director:	José María Sempere Luna

Resumen

La complementariedad de bases de Watson y Crick representa una gran fuente de estudio en biología estructural y las estructuras ahorquilladas (*hairpin structures*) son uno de los tipos de secuencia que más aplicaciones puede reportar. En particular, la inducción iterada de horquilla en una secuencia y su posterior elongación es una técnica que está siendo implementada para la amplificación e identificación de secuencias de DNA o RNA que se encuentren menos representadas en una muestra, con el fin de establecer su presencia en una situación genética determinada.

Sin embargo, este tipo de operaciones carece de herramientas que las modelen y que puedan servir de base a un investigador para seguir estudiándolas. Por ello, en este trabajo se realiza el análisis y la implementación de un analizador de secuencias biológicas que realice tanto el estudio estructural de la secuencia, el modelaje de una posible autoinducción iterada de las operaciones de horquilla-elongación-desnaturalización y la comparación entre dos secuencias diferenciadas únicamente en la introducción de un paso puntual de mutación.

Para el proyecto, se ha recurrido a la utilización de algoritmos estrictamente relacionados con la teoría de autómatas y lenguajes formales, que está ampliamente relacionada con la operación a modelar y nos permiten incorporar un diseño más eficiente a nuestro analizador.

El analizador se desarrollará en PYTHON y se pretende obtener una herramienta que trabaje en modo autónomo. Posteriormente, se evaluará su posible integración en un sitio web para su ejecución en red.

Palabras clave

Estructura ahorquillada, complementariedad de bases, teoría de lenguajes formales, extensión de horquilla, algorítmica matricial.

Abstract

Watson & Crick base-pairing represents a big source of study in structural biology and hairpin structures are one type of sequences with multiple applications to report. Particularly, the iterated induction of a hairpin in a sequence and the posterior elongation is a technique that is being implemented for the amplification and identification of DNA that is few represented in a sample.
However, this type of operations has not modeling tools that can be served as a base for an investigator. For this reason, in this project, the analysis and the implementation of a bio-sequence analyzer are carried out. It will perform the structural analysis of the sequence and the modeling of a possible iterated self-induction of three steps (self-hairpin, lengthening and denaturing) over a sequence.
In this project, we recurred to the utilization of algorithms strictly related with formal language and automaton theory, vastly related to the implemented operation. Moreover, these algorithms offer us the possibility of incorporating a more efficient design in our analyzer.
The analyzer will be developed in PYTHON and the objective is to obtain a tool working in stand-alone mode. Finally, a potential incorporation to a website will be assessed.

Key words

Hairpin structure, Base-pairing, Formal languages Theory, Hairpin Lengthening, Matritial algorithmics.

ÍNDICE GENERAL

Resumen.....I

1. Introducción.....1
1.1. Estructuras secundarias de DNA.....1
1.2. Evolución de estructuras ahorquilladas.....1
1.3. Teoría de autómatas y lenguajes formales.....3
1.4 Recursos informáticos.....5

2. Objetivos.....6

3. Métodos.....7
3.1 Extensión de horquilla.....7
3.2. Definición formal de extensión de horquilla.....9
3.3. Flujo del algoritmo.....10
3.4.1. Estructuras ahorquilladas.....11
3.4.2. Extensión de horquilla.....11
3.4.2.1. Reconocimiento de secuencias pertenecientes al lenguaje.....12
3.4.2.1.1. Matrices estructurales primarias.....12
3.4.2.1.2. Matriz M reconocedora de secuencias de $HL_k(w)$.....14
3.4.2.2. Cálculo de la distancia de extensión de horquilla.....16
3.4.2.2.1. Matrices estructurales secundarias.....16
3.4.2.2.2. Matriz H calculadora de la distancia.....17
3.4 Modo de uso de la herramienta.....19

4. Resultados y discusión.....20
4.1. Resultados de la aplicación.....20
4.2. Uso de bases de datos de estructuras ahorquilladas.....20

5. Conclusión y perspectivas de trabajo.....23

6. Bibliografía.....24

INTRODUCCIÓN

Estructuras secundarias de DNA

Además de la doble hélice canónica, las secuencias biológicas son capaces de plegarse en otras estructuras secundarias tanto intra como intermoleculares y adoptar una amplia variedad de formas basadas en motivos particulares y en interacciones con diversas proteínas. Formas estructuradas de DNA poseedoras de complementariedad son generadas en múltiples procesos celulares y están envueltas en funciones biológicas. Una de estas formas son las cadenas de DNA simples (ssDNA) producidas durante la replicación, la conjugación bacteriana, la transformación natural o por infecciones bacterianas. Estas cadenas de hebra simple son capaces de plegarse sobre sí mismas si poseen entidades moleculares (motivos, segmentos, repeticiones invertidas, etc.) a lo largo de su longitud que cumplan con la complementariedad de Watson y Crick. Cuando se produce este plegamiento la cadena forma una horquilla, y pasa a denominarse ahorquillada o auto-horquillada (Figura 1).

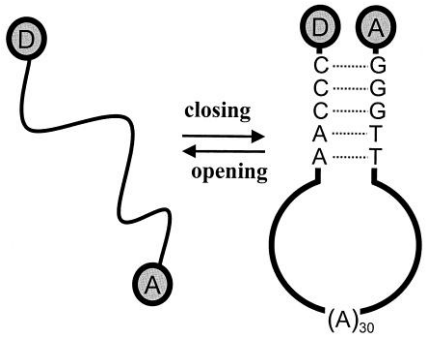

Figura 1: Secuencia en sus dos estados de cadena abierta simple o ahorquillada. (Wallace MI et al. 2001)

La existencia de horquillas en una secuencia determinada otorga a ésta múltiples funciones que en cadena simple no podría realizar. Así, las horquillas tienen una aplicación directa a la hora de iniciar la replicación de ciertas estructuras tanto lineales como circulares (Kornberg and Baker. 1992, Kahn. 2005), en la iniciación de la replicación bacteriana y en eventos de *splicing* y de recombinación. Además, las estructuras ahorquilladas pueden interaccionar con proteínas y modificar la fisiología de la célula mediante estructuras ahorquilladas cruciformes que modifican el estado enrollado del DNA (Pruss and Drlica. 1989), y mediante la inhibición de un sitio de unión en el DNA. Además, ciertas proteínas solo pueden reconocer y unirse específicamente a horquillas (Barabas et al. 2008). Por último, una de las últimas investigaciones con miRNAs ha demostrado que el origen y evolución de agrupaciones de estas entidades se produce por la formación *de novo* de nuevas horquillas *(Modelo de nueva horquilla)* en transcritos de miRNA ya existentes (Marco et al. 2014). Todas estas funciones y posibilidades podrán ser exploradas con nuestro analizador. Las estructuras secundarias ahorquilladas han sido ampliamente estudiadas en biología y son muchos los programas que reconocen horquillas y nos devuelven la estructura de la secuencia, tales como RnaFold, CoFOLD o ARAGORN (Laslett and Canback. 2004, Proctor and Meyer. 2013). Sin embargo, durante este proyecto, no solamente nos hemos centrado en el estudio de la estructura ahorquillada única, sino también en su evolución.

Evolución de estructuras ahorquilladas

El siguiente paso en nuestro estudio es definir el concepto de evolución de horquilla, anteriormente introducido. Cuando nos encontramos una estructura ahorquillada en algún punto de una secuencia, definiremos su evolución como la transformación de dicha secuencia en otra tras la aplicación

iterativa de una operación definida en la cual la propia formación de horquilla sirva como elemento evolutivo.

El estudio de la evolución de secuencias primarias hacia estructuras secundarias ahorquilladas y su iteración no es un concepto experimental, por lo que se debe recurrir a modelos matemáticos para estudiarlo y la teoría de autómatas y lenguajes formales (que será introducida más adelante) es, en este aspecto, de alta utilidad. Además, el estudio de este tipo de operaciones no es un concepto nuevo en teoría de lenguajes formales, sino que son varios los lenguajes motivados por la complementariedad de bases de Watson y Crick que han sido estudiados para secuencias biológicas.
En concreto, el estudio de este tipo de lenguajes empezó con una serie de artículos publicados por primera vez en 2006 (Bottoni et al. 2006) en los cuales se empezaron a definir los conceptos básicos de la teoría de lenguajes para las secuencias ahorquilladas. Los siguientes trabajos trataron de definir y modelar las primeras operaciones basadas en la complementariedad (Cheptea et al. 2006), denominadas completación de horquilla y reducción de horquilla (Manea. 2007, Manea et al. 2009).
Estas operaciones comenzaron a definir lenguajes sobre secuencias que se podían convertir en ahorquilladas sobre sí mismas y las cuales se asumían de estructura preservada, es decir, las horquillas producidas eran estables sin necesidad de elongación. Una vez producida la horquilla, se procedía a elongar la cadena hasta el final de su longitud (compleción de la horquilla) o bien a eliminar de la secuencia las bases que formaban parte de su longitud (reducción de horquilla). Las esquematizaciones de ambas operaciones se representan en la Figura 2. Si posteriormente a estas operaciones añadíamos un evento de desnaturalización de la doble cadena, obteníamos

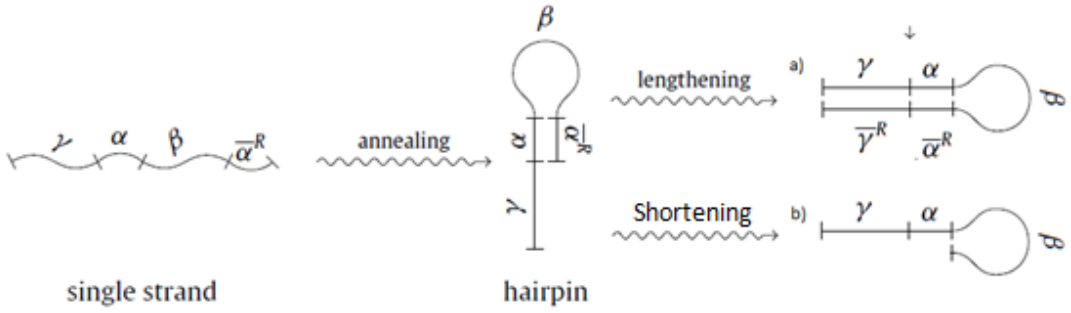

Figura 2: Esquemas de las operaciones de compleción de horquilla (a) y reducción de horquilla (b)

Sin embargo, estas primeras aproximaciones describían únicamente un paso del proceso, por lo que el siguiente paso fue definir las mismas versiones pero iterativas (Manea 2010), es decir, repitiendo la operación tantas veces como se deseara para formar secuencias nuevas.

Así, siguiendo los pasos de estas investigaciones, la motivación de nuestro proyecto era poder combinar los lenguajes ya existentes y dotar a nuestro analizador de características más biológicas y asumiblemente más ventajosas para ser implementadas computacionalmente con el objetivo de que pueda llegar a ser usado por los investigadores. En este punto, los desarrolladores de las primeras operaciones empezaron a definir un nuevo lenguaje que definía la operación de extensión de horquilla, un lenguaje que pretendía estudiar la evolución de una secuencia hacia otra más larga mediante auto-horquillado, extensión y desnaturalización de la misma, usando el mismo principio que la compleción de horquilla, pero variando la forma en que se podía extender la cadena después del horquillado. La forma de extensión dejaba de consistir en la completación de la secuencia ya que podría llegar hasta un punto intermedio si fuera necesario o interesante. Nuestro analizador partirá de estos diseños algorítmicos para realizar una serie de cálculos referentes a la extensión de horquilla, la operación que dará lugar a la evolución de una estructura autohorquillada.

No obstante, la evolución de las estructuras ahorquilladas hacia estructuras más complejas no ha sido objeto de estudio debido a la escasa implementación técnica y experimental existente y es por

ello que en este trabajo se explorarán, de manera básica, diversas vías de modelado computacional de operaciones complejas que involucran a esta evolución de horquillas.

Incluso sin ser operaciones implementadas, en la naturaleza es frecuente observar modelos que simulan operaciones de este tipo. Así, la operación de extensión de horquilla involucra a su vez operaciones más sencillas, como el auto-horquillado de secuencias, presente en algunas clases de virus como los Parvovirus, que tienen sus extremos 3' repetidos e invertidos para poder iniciar su replicación (Salzman and Fabisch. 1979), o la adición de nucleótidos a una secuencia para hacerla evolucionar a otra diferente, operación que se observa, con algunos matices como la introducción de cebadores externos, en las técnicas de amplificación de DNA, tales como la amplificación isotérmica mediada por horquilla (en adelante *LAMP*) (Fakruddin et al. 2013).

Volviendo al hecho de la ausencia de un modelo biomolecular experimental pre-establecido, este tipo de procesos ha de ser investigado con modelos matemáticos, como la teoría de autómatas y lenguajes formales, cuyos conceptos básicos definiremos a continuación.

Teoría de autómatas y lenguajes formales

En esta sección del trabajo, el objetivo principal será tratar de dar una introducción (Hopcroft and Ullman. 1979) razonablemente completa y formal acerca de las bases de la teoría de lenguajes formales, para conseguir así introducir los conceptos mínimos necesarios para entender el diseño y la implementación del analizador. Además, trataremos de relacionar los conceptos con las secuencias biológicas para demostrar la utilidad de esta teoría a la hora de definir de una manera eficiente cualquier cadena biológica.

Formalmente, un lenguaje (L) es un conjunto de palabras extraídas de un alfabeto, donde éste es denominado Σ y contiene un grupo de símbolos a partir de los cuales se puede crear una palabra siguiendo una serie de condiciones. Si el lenguaje comprende todas las palabras que pueden ser creadas a partir de Σ, el lenguaje se denomina universal y pasará a denotarse como Σ* y se definirá como:

$$\Sigma^* = \{a, b\}^* = \{\varepsilon, a, b, aa, ab, bb, \dots\}$$

Donde $a\ y\ b$ son los símbolos a partir de los cuales se pueden formar palabras del lenguaje y ε representa la cadena vacía, una palabra que no contiene ningún carácter y se puede obtener de todos los alfabetos. Si se está interesado en un lenguaje que no contiene la cadena vacía, a dicho lenguaje se le denominará Σ^+ y se definirá por:

$$\Sigma^+ = \Sigma^* - \{\varepsilon\}$$

En relación con estas definiciones, si queremos definir el alfabeto usado a la hora de definir una estructura genética, usaremos los símbolos definidos universalmente para los cuatro nucleótidos presentes en el DNA, es decir, adenina (a), citosina (c), timina (t) y guanina (g). El uso de estos símbolos viene dado por un criterio universal, aunque también son interesantes los alfabetos que contengan los símbolos de RNA o aminoácidos. En este sentido, el analizador actuará tomando como input secuencias en forma de DNA y empezará a trabajar con este tipo de secuencia. El alfabeto quedaría definido como:

$$\Sigma_{DNA} = \{a, c, t, g\}$$

Denominaremos longitud de una palabra al número de símbolos que contiene. Dada una palabra x y otra y, pertenecientes a Σ y un elemento de nuestro alfabeto $a \in \Sigma$, se define la longitud de x como:

$$|x| = \begin{cases} 0 \; si \; x = \varepsilon \\ |y| + \; 1 \; si \; x = ya \end{cases}$$

De un modo más descriptivo, dada una palabra $w = actgactg$, definiremos su longitud $|w| = 8$.
De la misma manera, es importante definir el concepto de segmento de palabra, es decir, un fragmento contenido en una palabra, de una longitud menor o igual. El segmento viene definido como $w[i..j]$, donde i y j representan los índices de inicio y fin del segmento dentro de la palabra. Así, $w[1..3] = act$.
A partir de la definición de un segmento, que de una forma informal define cualquier fragmento de la secuencia, podemos además incluir las definiciones de prefijo y sufijo.
Dadas las palabras $w, v \in \Sigma^*$, se define:

- w es un prefijo de v si existe una palabra u ∈ Σ^* con v = wu
- w es un sufijo de v si existe una palabra u ∈ Σ^* con v = uw

De acuerdo con el funcionamiento de nuestro analizador, se debe establecer una definición formal de concatenación de palabras, es decir, la adición de palabras a las ya existentes, que también se define en la teoría como:

Sean x y w dos cadenas tales que x, w ∈ Σ^*, se denomina concatenación de x y w a una nueva cadena xw constituida por los símbolos de la cadena x seguidos por los de la cadena w.

Una vez definidos los conceptos básicos de teoría de lenguajes que usará nuestro analizador y los símbolos que van a a componer dichas secuencias, el siguiente paso será la definición de una serie de conceptos de teoría de lenguaje aplicados a la formación de horquillas, esenciales en nuestra herramienta.

Por convención, las cadenas de DNA están descritas con los elementos del alfabeto Σ con orientación 5' a 3' y esta será la direccionalidad de la secuencia en la herramienta. Una involución sobre un elemento de una cadena se define como una aplicación de un elemento en otro, es decir, un elemento es reconocido como otro durante el análisis de la secuencia. Las cadenas de DNA cumplen estas involuciones gracias a sus bases complementarias. Sea la notación $\bar{.}$, las involuciones quedarían de la siguiente manera:

$$\bar{g} = \; c \; ; \; \bar{c} = \; g; \; \bar{t} = \; a; \; \bar{a} = \; t$$

De una manera general, una involución sobre un conjunto S se define como un mapa biyectivo:

$$\sigma: S \to S, tal \; que \; \sigma = \sigma^{-1}$$

De esta forma, decimos que cada uno de nuestros elementos tiene una complementariedad, es decir, los pares **c-g** y **a-t** son complementarios.

Un anti-morfismo se define como una secuencia reversa y nos sirve para introducir las cadenas reversas que van a buscarse durante el algoritmo. Sea la notación ^{R}, se define un anti-morfismo como:

$$(a_1 a_2 \dots a_n)^R = a_n \dots a_2 a_1$$

Para entender de una manera práctica estas definiciones, se propone el siguiente ejemplo:

$$w = atgc$$
$$\overline{w} = tacg$$
$$w^R = cgta$$
$$\overline{w}^R = gcat$$

En lenguajes iterados y concatenados, como el caso que nos ocupa, es importante definir cada iteración mediante una operación de cierre. En este caso, existen dos tipos de cierre, el cierre estrella y el cierre positivo.

El cierre estrella define al lenguaje cuyas palabras son todas las que se pueden obtener realizando 0 o más concatenaciones de palabras del lenguaje (L). En nuestro caso, las palabras de L serán todas aquellas secuencias que tenemos en el análisis pero también sus involuciones y anti-morfismos formados siempre con símbolos de nuestro alfabeto. El cierre estrella se define como:

$$L^* = \bigcup_{i \geq 0} L^i$$

Así, en resumen, nuestro lenguaje buscará estructuras formadas a partir de símbolos de nuestro alfabeto, en una estructura definida, realizaremos las interacciones adecuadas sobre ellas, incluyendo sus involuciones según la complementariedad de Watson y Crick y finalizaremos cada iteración con un cierre para pasar a la siguiente.

Recursos informáticos

El analizador se realizará con el lenguaje Python, uno de los llamados lenguajes de alto-nivel, que nos ofrece facilidades a la hora de escribir el código y leerlo computacionalmente, con lo que la eficiencia de los algoritmos es mayor. Python ofrece también un manejo más sencillo a la hora de trabajar con cadenas de DNA y de identificar las involuciones y los anti-morfismos, así como una mayor sencillez a la hora de hacer evolucionar dichas secuencias de DNA, que es el objetivo de nuestro trabajo. Además, este lenguaje ha sido ampliamente usado en muchas aplicaciones de la teoría de lenguaje, como gramáticas génicas (*gene grammars*) o el estudio del lenguaje natural y la búsqueda de patrones dentro de secuencias.

El analizador correrá mediante comandos en un terminal añadiendo una serie de parámetros que serán necesarios para el funcionamiento de nuestra operación. Tanto estos parámetros como la definición de extensión de horquilla se definirán a continuación.

La herramienta ha sido programada y testada en un sistema operativo Ubuntu LTS 16.04 (Linux x64) y tanto su disponibilidad como la de los módulos externos al lenguaje se ofrecen bajo demanda.

OBJETIVOS

Dada la poca investigación en lenguajes que simulen la evolución de estructuras ahorquilladas simples en otras más complejas de manera iterada, en este trabajo, se plantea como objetivo principal el análisis y la implementación de una herramienta analizadora que simule una operación de este tipo, llamada extensión de horquilla.

Aunque esta operación sea potencialmente aplicable en algunas técnicas biológicas experimentales, la meta es implementar un algoritmo que modele la operación de manera eficiente. En un futuro, se valorará la incorporación de elementos externos a este algoritmo con el fin de modelar dichas técnicas experimentales.

El analizador tomará una secuencia X, una secuencia W y una longitud de horquilla (k) como inputs y cumplirá los siguientes objetivos:

- Dada la longitud k, la herramienta mostrará todas las horquillas de esta longitud que se pueden dar en la secuencia X. Esta operación está pensada para dar información estructural en la secuencia, pudiendo ver de una manera sencilla si es posible la aparición de una horquilla en una posición determinada de la secuencia o la secuencia de nucleótidos que quedarían dentro de la horquilla.
- Dadas las secuencias X y W, y dada la longitud k, la herramienta modelará el proceso de evolución de la secuencia X a W, es decir, mostrará como W puede ser formada a partir de X, mediante la operación de extensión de horquilla, que involucrará tres procesos ordenados llamados auto-horquillado, extensión y desnaturalización. La operación de auto-horquillado se producirá hibridando tantas bases como la longitud de horquilla k.
- Además del modelado de la evolución, la herramienta nos informará sobre la distancia de extensión de horquilla respecto a k, que se define como el número de veces que se deben repetir las tres operaciones anteriores para pasar de X a W.

MÉTODOS

Extensión de horquilla

El siguiente paso en nuestro estudio es definir el concepto de evolución de horquilla (Manea et al. 2010), anteriormente introducido. Cuando nos encontramos una estructura ahorquillada en algún punto de una secuencia, definiremos su evolución como la transformación de dicha secuencia en otra tras la aplicación de una operación definida relacionada con horquillas.
En nuestro caso, la operación analizada se denomina extensión de horquilla y consiste en pasos sucesivos de tres sub-operaciones: Auto-horquillado → Extensión de secuencia → Desnaturalización. Las tres sub-operaciones se representan en la Figura 3.

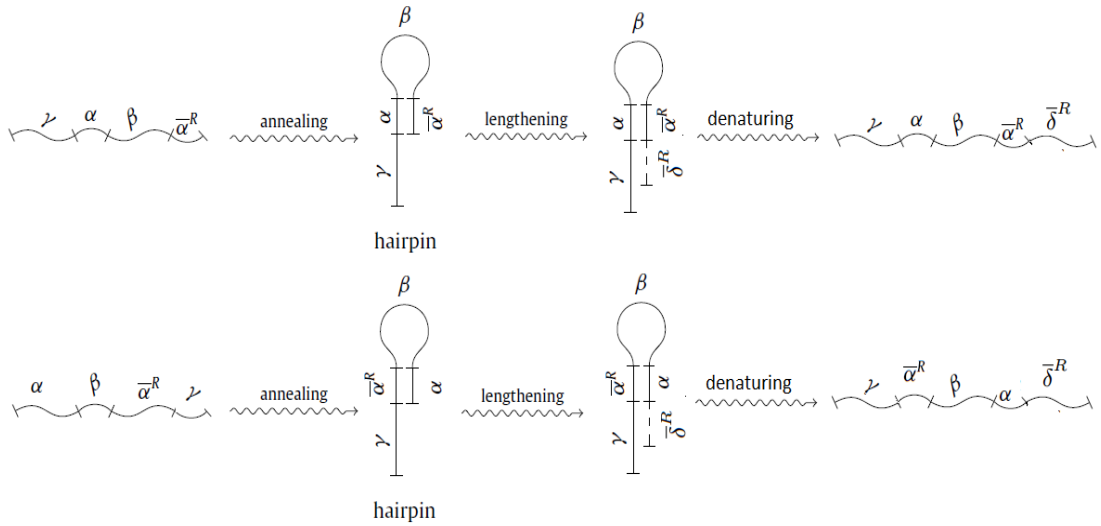

Figura 3: Esquema de las tres etapas de una iteración de extensión de horquilla por ambos extremos de la secuencia.

Como se observa en la figura 3, el auto-horquillado se define como la inducción de una horquilla en una secuencia de cadena simple. El extremo final de la cadena (α), de una longitud k determinada, hibrida con un fragmento contenido en la secuencia complementario y reverso a $\alpha(\overline{\alpha^R})$. La extensión es el alargamiento de la secuencia en el cual se introduce un número determinado de nucleótidos complementarios al fragmento de cadena comprendido entre el final de la horquilla y el extremo de la secuencia que no se había auto-horquillado. Cabe destacar aquí que el algoritmo predecirá solamente la elongación de toda la secuencia restante (γ) en todas las iteraciones menos en la última, donde la elongación de la secuencia podrá ser parcial (δ^R). Este proceso representa una ventaja biológicamente hablando, ya que las posibilidades para evolucionar la secuencia primera aumentan exponencialmente, pudiéndose modelar un mayor número de procesos y posibilidades de manera sistemática. El paso de la desnaturalización se define como la ruptura de los puentes de hidrógeno entre las bases complementarias gracias a la cual la cadena vuelve a su forma simple, a partir de la cual se repite la operación de extensión de horquilla.

Así, nuestro analizador tomará como input dos secuencias dadas, denominadas ***X*** y ***W***. La operación de extensión de horquilla será producida de manera iterada sobre ***X*** con el fin de calcular la distancia, si existe, entre ***X*** y ***W***. La definición básica de distancia se define como el número mínimo de iteraciones de extensión de horquilla necesarias para pasar de ***X*** a ***W.*** Tanto la operación como el lenguaje que la define y la distancia se definirán de una manera más formal posteriormente.

En este punto es importante definir una serie conceptos biológicos sin los cuales la operación quedaría parcialmente indefinida y que el usuario de la aplicación debe tener en cuenta.

1. Longitud de horquilla: Se define como longitud de horquilla, ***k,*** la longitud del segmento de secuencia que forma la horquilla, es decir, la longitud de las secuencias hibridadas por complementariedad. El valor de ***k*** podrá ser definido por el usuario de la herramienta pero dicho valor será el mismo en cada iteración.
2. Repeticiones invertidas (IRs): También llamadas palíndromos. Un par de secuencias IR es un par de secuencias complementarias y reversas de otra comprendida a lo largo de la misma secuencia. Este tipo de secuencias son las que forman la horquilla.
3. Direccionalidad de la polimerasa: Debido a posibles aplicaciones no biológicas, el algoritmo está preparado para reconocer horquillas producidas a partir de ambos extremos, tanto 3' como 5'. No obstante, la elongación de la horquilla formada se produce de un modo directo, sin fragmentos de Okazaki a lo largo de la secuencia. Por tanto, es importante conocer bien las secuencias a la hora de calcular su distancia, pero también lo es conocer la direccionalidad de la polimerasa que se usaría, ya que solo así se modelaría la operación de manera óptima.
4. Secuencias repetidas: Este tipo de secuencias se definen como ocurrencias múltiples de un fragmento dentro de una misma secuencia. En este caso, las repeticiones son una de las aplicaciones potenciales de nuestro analizador, ya que es su existencia la que hace que las secuencias que puedan llegar a formarse mediante extensión de horquilla sean diferentes. El algoritmo estará preparado para reconocer dichas repeticiones en caso de ser necesario, pero calculará la distancia en base únicamente a la secuencia x óptima que se pueda transformar en w.

Por tanto, la funcionalidad del analizador residirá en el estudio de las estructuras ahorquilladas de una secuencia y en el modelaje de la operación de extensión de horquilla. No obstante, la definición no es completa si no otorgamos a nuestro analizador un contexto biológico. En otras palabras, en este punto es importante caracterizar qué fenómenos naturales o técnicas implementadas ya existentes podríamos modelar.

Con la funcionalidad de estudio de la estructura, el analizador será capaz de dar al investigador diversos datos sobre la secuencia, tales como posibles sitios de recombinación y la secuencia de una horquilla, pero también será posible estudiar, en caso de que sea adecuado, si la secuencia a estudiar produce horquillas en sus extremos. En este sentido, si la secuencia en cuestión que se está analizando es un cebador, la existencia de una horquilla en sus extremos podría significar que en una PCR estos cebadores no dieran ninguna clase de amplificación (problema llamado *back-hybridation*).

Por otra parte, con la funcionalidad de modelaje, todavía no existe ninguna técnica experimental diseñada para esta operación, por lo que la búsqueda de aplicaciones se dificulta. Aunque existen modelos de computación que estudian este tipo de operaciones con secuencias genéticas, como la WPCR (Reif and Majmuder. 2008) y sus derivadas, como PWPCR, mediada por PNAs, a nivel de laboratorio, la implementación de una técnica que simule la extensión de horquilla y su elongación parece ser difícil debido a la inestabilidad de las secuencias auto-horquilladas. No obstante, existen técnicas que parten de una secuencia dada y mediante la utilización de cebadores externos, sí que consiguen obtener una estructura auto-horquillada estable y que pueda derivar en otra secuencia. Un ejemplo de la aplicabilidad de estas técnicas se encuentra en las técnicas de amplificación de DNA ajenas a la técnica de PCR. El ejemplo más claro es la *LAMP*, que tiene por productos unas estructuras con múltiples bucles y repeticiones invertidas.

Lo que se pretende con este analizador es obtener una primera aproximación que calcule la evolución de la secuencia X a W mediante extensión de horquilla para así poder modelar, en un futuro, aplicando los principios de las técnicas de amplificación, evoluciones de las estructuras ahorquilladas hacia otras con unas características modificadas concretas que mejoren las que ya tiene una secuencia, tales como duplicación de sitios de unión, dominios que le confieran estabilidad o protección, o incluso para poder llegar a crear estructuras interesantes para modelar nanopartículas basadas en DNA.

Esta definición, entendible desde el punto de vista biológico, no se puede considerar una definición formal y sujeta a la teoría de lenguajes de la extensión de horquilla, por lo que la definición bajo este último punto de vista se describe a continuación.

Definición formal de extensión de horquilla

Sea V un alfabeto, para cualquier palabra $w \in V^+$, definimos el lenguaje de extensión de horquilla de w en función de k, denotada por $HL_k(w)$, para una $k \geq 1$, como sigue:

$$HLP_k(w) = \{\delta^R w | w = \alpha\beta\overline{\alpha^R}\gamma, |\alpha| = k, \alpha, \beta, \gamma \in V^+ \; y \; \delta \; es \; un \; prefijo \; de \; \gamma\}$$
$$HLS_k(w) = \{w\delta^R | w = \gamma\alpha\beta\overline{\alpha^R}, |\alpha| = k, \alpha, \beta, \gamma \in V^+ \; y \; \delta \; es \; un \; sufijo \; de \; \gamma\}$$
$$HL_k(w) = HLP_k(w) \cup HLS_k(w)$$

Con esta definición, observamos que nuestro lenguaje estará formado a su vez por la unión de dos lenguajes diferentes, según se produzca la horquilla desde un extremo o desde el otro. Esto nos informa de que, aunque biológicamente hablando las horquillas solo van a tener sentido si la polimerasa puede extender en su dirección habitual, el algoritmo del lenguaje estará preparado para identificar horquillas producidas en ambos extremos.
Para optimizar la identificación de elementos, definimos ciertos símbolos que delimitan fragmentos de la secuencia, como se puede ver en la figura 4 y que se explican a continuación:

- $\alpha \; y \; \overline{\alpha^R}$, son las secuencias IR, sobre las cuales se asienta la horquilla. Las secuencias delimitadas por estos símbolos son complementarias reversas (involución y anti-morfismo en la introducción).
- β es la secuencia que queda dentro de la horquilla y que es de cadena simple, sin hibridar ninguna de sus bases.
- γ es el resto de la secuencia que participa en el horquillado y que no pertenece a ninguno de los símbolos anteriores.
- δ^R es la secuencia que se extenderá en el segundo paso del proceso. Esta cadena, por definición será un fragmento reverso complementario de un fragmento de longitud cualquiera entre $1 \leq |\delta^R| \leq |\gamma|$.

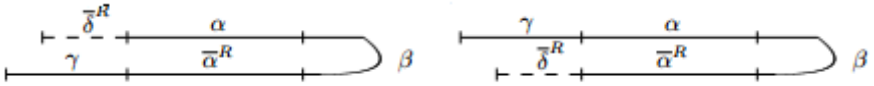

Figura 4: Elementos de la extensión de horquilla. HLPk (w) (Izq.) y HLSk (w) (Der.)

Además, es importante tener en cuenta que las cadenas $\alpha, \beta, \overline{\alpha^R}, \gamma \; y \; k \in V^+$, lo cual significa que estas cadenas no pueden estar vacías, es decir, que al menos deben contener un símbolo. Esta equivalencia nos indica que en nuestro algoritmo dos bases adyacentes no podrían formar

una horquilla entre ellas, ya que la cadena β estaría vacía, y que la horquilla no puede darse entre secuencias IR del principio y del final de la secuencia. El algoritmo está preparado para este tipo de situaciones.

El otro concepto importante que se detalla es el de k, que se define como la longitud de la horquilla. Nótese que la definición opera en función de k y que ésta será definida por el usuario. Se extiende pues el valor de k a cualquier valor, tal que $HL_{k+1}(\mathrm{w}) \subseteq HL_k(w)$, para cualquier $w \in V^+ y\ k \geq 1$.

Por último, la definición de extensión de horquilla extendida a lenguajes queda como:

$$HL_k(\mathrm{L}) = \bigcup_{w \in L} HL_k(w)$$

La versión iterada de la extensión de horquilla se define como:

$$HL_k^0(w) = \{w\}, HL_k^{n+1}(w) = HL_k\left(HL_k^n(w)\right), HL_k^*(w) = \bigcup_{n \geq 0} HL_k^n(w)$$

$$HL_k^*(L) = \bigcup_{w \in L} HL_k^*(w)$$

FLUJO DEL ALGORITMO

El analizador que presentamos tendrá dos funcionalidades distintas. Por un lado, buscaremos todas las secuencias ahorquilladas que podemos encontrar a lo largo de una secuencia determinada y por otro lado, podremos calcular la distancia desde una secuencia a otra mediante la operación de extensión de horquilla y modelar dicha evolución. A continuación, en la figura 5, se muestra un esquema general del proceso del algoritmo.

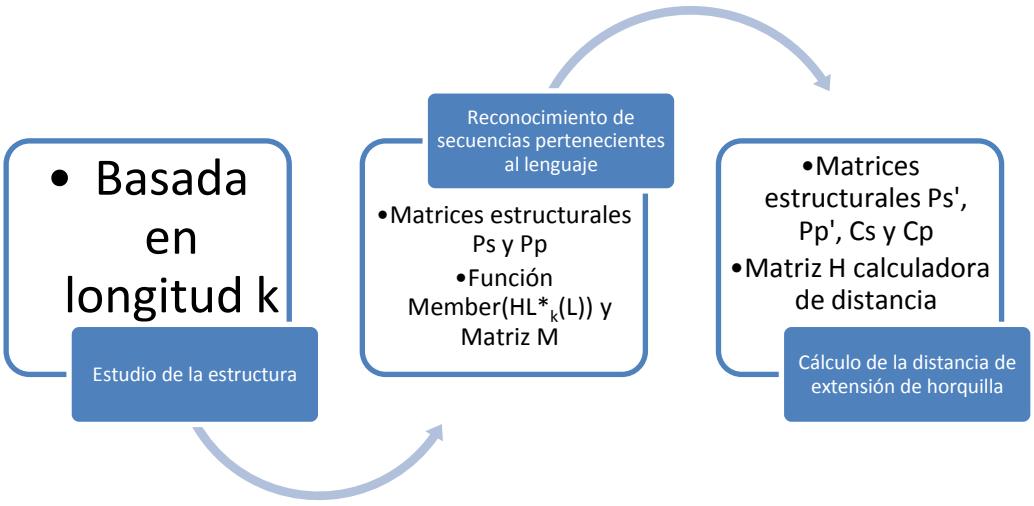

Figura 4: Representación general del flujo de la herramienta

ESTRUCTURAS AHORQUILLADAS

En esta primera aproximación, se debe destacar que la finalidad de esta operación es definir la estructura de la secuencia de una manera simple, bien para ver si el diseño de una secuencia que quiere ser usada como cebador está bien realizado o bien para definir si la molécula es capaz de auto-horquillarse para servir como centro de unión para otras moléculas.

Nótese que nuestro algoritmo solo buscará una única horquilla a lo largo de la secuencia, es decir, no contemplará la búsqueda de pseudonudos o estructuras con múltiples horquillas entrelazadas, como los tRNAs, y dicha horquilla será siempre de la longitud introducida por comando.

Además, la definición de la estructura vendrá determinada por la longitud de horquilla, es decir, por el número de bases que definen la complementariedad.

EXTENSIÓN DE HORQUILLAS

La utilidad de este algoritmo residirá en su eficiencia y, por tanto, al plantear el problema de la extensión de horquillas, pensar en una aproximación que recorra nuestras secuencias objeto de estudio varias veces para encontrar las reversas y reproducir su elongación no tiene sentido, por lo que habremos de estructurar los datos de una manera más ordenada y sencilla. En un primer momento se podría pensar que el esquema de trabajo debería consistir en buscar la complementariedad del extremo en la misma secuencia y posteriormente, gracias a la definición de la complementariedad, elongar las cadenas con sus correspondientes bases, pero no hay que olvidar que la operación que estamos definiendo no contempla la elongación de un número de bases concreta, y ni tan solo que el número de bases comprendidas en la elongación haya de ser el mismo en cada iteración, sino que este número puede variar en cada paso, de ahí la originalidad en el estudio de esta operación. Por tanto, si tomáramos esta estrategia, la cadena se tendría que recorrer varias veces, elongando cada vez un número de bases distinta hasta comprender todas las posibilidades, estableciendo así ciertos problemas en cuestiones de computación, haciéndola lenta y poco eficiente, por lo que la estructuración de los datos debe hacerse de manera sencilla y a la vez fiable, con el fin de rebajar esta tasa polinómica del algoritmo y hacerlo computable en términos de eficiencia.
En este proyecto, exploramos una vía de estructuración de datos distinta, denominada algorítmica matricial, y consistente en la utilización de matrices, en las cuales cada posición representará un índice de la secuencia determinado, es decir, la posición en que se encuentre un nucleótido determinado que nos pueda interesar por diversas razones, como por ser la posición donde empiece o termine la complementariedad para formar la horquilla, o por ser el final o el principio de la elongación que se produce en cada iteración. Así, la cadena se tendrá que recorrer únicamente al principio de la estrategia, establecerá los índices y calculará la distancia.

Nótese que el algoritmo debe estar preparado para producir una operación hacia ambos lados de la secuencia, aunque biológicamente ha quedado remarcado que la extensión de horquillas dependerá de la direccionalidad de la polimerasa, y que, por tanto, será necesaria la utilización de dos matrices, una por cada extremo. De esta forma, los índices tendrán una funcionalidad recursiva y en pasos posteriores podremos realizar la implementación basándonos en ellos, sin necesidad de ir recorriendo la cadena, es decir, simplemente realizando una comparación de índices, cosa que computacionalmente hablando nos otorga mayor eficiencia.

RECONOCIMIENTO DE SECUENCIAS PERTENECIENTES AL LENGUAJE

MATRICES ESTRUCTURALES PRIMARIAS

El primer objetivo por tanto, de nuestra estrategia será diseñar dos matrices directrices que guíen el algoritmo hasta obtener el resultado de la distancia. Al hablar de estructuras ahorquilladas y extensión de horquillas, el primer paso debe ser siempre la búsqueda de estructuras complementarias, a partir de las cuales se obtiene nuestra horquilla. Por eso, el algoritmo recorrerá todas las palabras de longitud de horquilla *k*, que está definida por el usuario y buscará sus complementarias reversas tanto a su izquierda como a su derecha, y almacenará los índices donde empiecen y acaben estas ocurrencias.
Estas matrices, una por cada extremo de la secuencia, nos ayudarán a obtener las secuencias que pertenecen a nuestro lenguaje $HL_k^*(L)$ y a calcular la distancia final. Las definimos como dos matrices Ps y Pp triangulares superiores de dimensiones n x n, siendo n la longitud de la secuencia final w. Las entradas de la matriz serán números naturales con el siguiente significado:

- Ps[i][j] almacena la posición en la cual empieza la ocurrencia más a la derecha de $\overline{w[i][j]}^R$ en el fragmento w[1..i-1]. Por defecto, asumimos que Ps[1][j] = 0 para toda j≤ n, ya que el fragmento w[1..i-1] en este caso no existe, y Ps[i][j] = 0 para toda i<j≤n, ya que en este caso solo nos interesa una dirección de la cadena, de izquierda a derecha. No se analiza el otro sentido.
- Pp[i][j] almacena la posición en la cual empieza la ocurrencia más a la izquierda de $\overline{w[i][j]}^R$ en el fragmento w[j + 1..n]. Por defecto, asumimos que Pp[i][n] = 0 para toda i ≤ n, ya que el fragmento w[j + 1..n] en este caso no existe, y Pp[i][j] = 0 para toda j<i≤n, ya que en este caso solo nos interesa una dirección de la cadena, de izquierda a derecha. No se analiza el otro sentido.

Cabe destacar que el algoritmo está diseñado en este caso para reconocer solamente la ocurrencia más a la derecha del fragmento w[1..i-1] y la más a la izquierda del fragmento w[j+1..n], es decir, en el caso de que haya más de una ocurrencia del fragmento w[i..j] en cualquiera de los extremos el algoritmo solo produciría elongación a partir de las ocurrencias acabadas de definir. Este hecho ocurre porque en una matriz solamente se puede almacenar una posición. Una posible solución a este problema, que no se ha implementado en el analizador, sería la de crear una matriz estructural por cada ocurrencia.
Para llenar la matriz recurriremos a las siguientes ecuaciones:

$$Ps[i][j] = i - LO_{[w[i..n],\overline{w[1..i-1]}^R}[j-i+1]\ para\ todo\ i\ y\ j\ tales\ que\ n \geq j \geq i > 1$$
$$Pp[i][j] = j + LO_{\overline{w[1..j]}^R,w[j+1..n]}[j-i+1]\ para\ todo\ i\ y\ j\ tales\ que\ n > j \geq i \geq 1$$

Así, i y j representan los índices a partir de los cuales vamos a buscar las ocurrencias, por lo que son el primer factor a tener en cuenta. El siguiente implicado es el índice $LO_{x,y}$ (Leftmost Ocurrence), definido como:

$$LO_{x,w}[y] = \begin{cases} t, si\ la\ LO\ de\ x[1..y] en\ w\ se\ encuentra\ en\ w[t-y+1..t] \\ 0, si\ x[1..y]\ no\ aparece\ en\ w \end{cases}$$

Los términos LO de esta definición pertenecen a un algoritmo de búsqueda de patrones de una cadena en otra en tiempo lineal $\mathcal{O}(|x| + |w|)$ denominado algoritmo KMP (Cormen et al. 1990). Según la definición, la manera de usar el algoritmo involucra tres elementos principales: x, w y el índice y.

Respecto a las secuencias x, es decir, las secuencias que se buscarán en w, nuestro algoritmo las define para cada matriz como segmentos de la secuencia que se buscan o bien en el fragmento anterior o bien en el posterior. Nótese que en Ps se busca la secuencia natural mientras que en Pp se busca la secuencia complementaria reversa para así cumplir con lo establecido en el algoritmo KMP, es decir, de lo que se trata es de buscar la LO con las condiciones del algoritmo, y únicamente de esta manera se cumplen.

Las secuencias w son los fragmentos de secuencia donde buscaremos las x y se corresponden con los fragmentos anteriores y posteriores al fragmento que se esté estudiando.

Por último, el índice y es una longitud de cadena. Esta longitud corresponde con la longitud que tomaremos de x, y se fija a través de los valores i y j del segmento que queremos estudiar. Así, si estamos estudiando el fragmento w[i..j] = w[5..7], deberemos definir una $|w[i..j]| = 3$, y así para todas las posibles longitudes, por lo que de una manera general, la expresión para calcular y será $y = j - i + 1$.

Así pues, para el cálculo de Ps y Pp tomaremos un segmento de secuencia definido por i y j, buscaremos su LO a ambos extremos, realizando las operaciones complementarias y reversas necesarias y almacenaremos la posición correspondiente, todo ello mediante un algoritmo eficiente que realizará la operación en un tiempo lineal.

A continuación se muestra el algoritmo que calcula Ps y Pp en la figura 5.

Algorithm 1 $ComputeMat(w)$: returns the values of the two matrices

1: **for** $i = 2$ to n **do**
2: Compute $LO_{w[i..n],\overline{w[1..i-1]}^R}$;
3: **end for**
4: **for** $j = 1$ to $n - 1$ **do**
5: Compute $LO_{\overline{w[1..j]}^R,w[j+1..n]}$;
6: **end for**
7: **for** $i = 1$ to n **do**
8: **for** $j = 1$ to n **do**
9: **if** $i = 1$ or $j < i$ **then**
10: $P_s[i][j] = 0$
11: **else**
12: $P_s[i][j] = i - LO_{w[i..n],\overline{w[1..i-1]}^R}[j - i + 1]$
13: **end if**
14: **if** $j = n$ or $j < i$ **then**
15: $P_p[i][j] = 0$
16: **else**
17: $P_p[i][j] = j + LO_{\overline{w[1..j]}^R,w[j+1..n]}[j - i + 1]$
18: **end if**
19: **end for**
20: **end for**
21: Return the pair of matrices (P_s, P_p)

Figura 5: Algoritmo calculador de las matrices estructurales primarias

MATRIZ M IDENTIFICADORA DE SECUENCIAS

Una vez calculadas las matrices estructurales, el siguiente paso de nuestra estrategia consistirá en determinar si la palabra que estamos analizando pertenece o no a nuestro lenguaje $HL_k^*(L)$. El algoritmo implementado con esta función es capaz de computar una matriz de dimensiones n x n con entradas binarias definidas por la expresión $M[i][j] = (w[i..j] \in HL_k^*(L))$, lo cual significa que la posición $M[i][j]$ tiene el mismo valor lógico o valor real que la expresión $w[i..j] \in HL_k^*(L)$. La computación de la matriz M está basada en una aproximación dinámica. Inicialmente, para todo $i, j \in \{1, \dots, n\}$, estableceremos una $M[i][j] = 1\ si\ w[i..j] \in L$ y una $M[i][j] = 0\ si\ w[i..j] \notin L$. En este punto, la única posición que podremos establecer como $M[i][j] = 1$ será aquella en la que se encuentre la palabra x, que es la única palabra que conocemos y que puede formar parte del lenguaje. Además, esta palabra x nos ayudará a iniciar el cálculo de la matriz. El algoritmo en este punto está preparado para buscar todas las ocurrencias de x en w, pero solo nos informará de la distancia a partir de la ocurrencia que sea capaz de transformarse en w a través de la operación de extensión de horquilla. Siguiendo con la computación, analizaremos todos los segmentos de w en orden creciente de longitud. En definitiva lo que vamos a conseguir con esta matriz es obtener todas aquellas palabras que sean de la forma definida anteriormente (ap. "Extensión de horquilla"), es decir, que pertenezcan a $HLS_k(w)$ o bien a $HLP_k(w)$. No obstante, con esto no es suficiente, sino que también debemos tener en cuenta que lo que buscamos no son palabras de esta forma sin más, sino que además buscamos que se formen a partir de x o sus derivadas de los eventos de extensión de horquilla (el término derivación se definirá en el apartado siguiente de manera formal) por lo que este hecho también deberá ser implementado en el analizador.
Siguiendo con el hecho de que deberemos identificar una forma concreta de secuencia que pertenezca a nuestro lenguaje, podemos ir un paso más allá y asegurar que un segmento de w puede ser obtenido por extensión de horquilla iterada definida por una longitud *k* si una de las siguientes condiciones se satisface:

- Existe un índice s tal que $i + 2k + 2 \leq s < j$ tal que $w[i..s] \in HL_k^*(L)$ y $w[i..j] \in HLS_k(w[i..s])$
- Existe un índice t tal que $i < t \leq j - 2k - 2$ tal que $w[t..j] \in HL_k^*(L)$ y $w[i..j] \in HLP_k(w[t..j])$

Como vemos, las ecuaciones nos indican que debemos buscar sub-segmentos del fragmento w[i..j], definidos por s y por t y que pertenezcan al lenguaje, es decir, que sea de cualquiera de las formas definidas anteriormente (ap. Definición formal de extensión de horquilla).

En cuestión de computación por tanto, deberemos buscar una serie de igualdades que nos permitan implementar correctamente la matriz. Después de obtener las matrices estructurales, podemos concluir que la matriz M en la posición $M[i][j] = 1$ siempre que se cumplan una de las siguientes condiciones:

$$w[i..j] \in L$$

En este caso, solo la palabra x se puede asegurar como perteneciente al lenguaje, ya que es a partir de la cual empezaremos el cálculo. Todas las palabras intermedias hasta llegar a w serán mayores en longitud que la palabra x.

$$Existe\ un\ índice\ s\ \ tal\ que\ i \leq s \leq j,$$
$$M[i][s] = \ 1\ y\ w[s-k+1..j] es\ un\ segmento\ de\ \overline{w[\imath..s-k]}^R$$

En este caso, nuestra búsqueda se centra en un índice s, intermedio del fragmento estudiado, que delimite una secuencia teniendo en cuenta la longitud de horquilla y que la secuencia complementaria reversa de la secuencia definida se encuentre posterior al primer índice del fragmento (índice i). En otras palabras calcularemos si se cumple la siguiente relación $Ps[s-k+1][j] \geq i$ y si se cumple, la matriz será igual a 1.

$$Existe\ un\ índice\ t\ \ tal\ que\ i \leq t \leq j,$$
$$M[t][j] = \ 1\ y\ w[i..t+k-1] es\ un\ segmento\ de\ \overline{w[\imath..t+k]}^R$$

En este caso, nuestra búsqueda se centra en un índice t, intermedio del fragmento estudiado, que delimite una secuencia teniendo en cuenta la longitud de horquilla y que la secuencia complementaria reversa de la secuencia definida se encuentre anterior al segundo índice del fragmento (índice j). En otras palabras calcularemos si se cumple la siguiente relación $Pp[i][t+k-1] \geq j$ y si se cumple, la matriz será igual a 1.

Para entender las inecuaciones anteriores hay que conocer el significado de s y t, que no son más que índices intermedios de w[i..j] en los cuales sabemos que o bien w[i..s] o w[t..j] pertenecen al lenguaje. Así, con la introducción de un array que varíe cada vez que encontremos una palabra que pertenezca al lenguaje, haremos que las igualdades M[i][s] = 1 y M[t][j] = 1 se cumplan y solo nos tengamos que preocupar por el fragmento complementario. La primera palabra que hará cambiar los índices será la palabra x.

A continuación se muestra el algoritmo que establece los valores de M y que decide si la cadena $w \in HL_k^*(L)$ en la figura 6.

Algorithm 2 $Member_{L,k}(w)$: returns the truth value of $w \in HL_k^*(L)$

```
1: Initialize matrix M: if w[i..j] ∈ L set M[i][j] = 1, otherwise set M[i][j] = 0
2: (P_s, P_p) = ComputeMat(w);
3: Initialize arrays r and l;
4: for len = 1 to n do
5:    for i = 1 to n − len + 1 do
6:       j = i + len − 1;
7:       if M[i][j] = 0 and r[i] ≠ 0 and P_s[r[i] − k + 1][i] ≥ i then
8:          M[i][j] = 1;
9:       end if
10:      if M[i][j] = 0 and l[i] ≠ 0 and P_p[i][l[j] + k − 1] ≤ j then
11:         M[i][j] = 1;
12:      end if
13:      if M[i][j] = 1 then
14:         r[i] = j and l[j] = i;
15:      end if
16:   end for
17: end for
18: Return true if M[1][n] = 1 or false otherwise.
```

Figura 6: Algoritmo calculador de la matriz M

CÁLCULO DE LA DISTANCIA DE EXTENSIÓN DE HORQUILLA

Como ya se ha definido anteriormente, la distancia entre dos palabras x y w se define como el número mínimo de pasos de extensión de horquilla, p, necesarios para que w sea obtenida de x. De una manera más formal, la distancia de extensión de horquilla, denotada como $HLD_k(x,w)$, quedaría como:

$$HLDk(x,w)\begin{cases} min\{p \mid x \in HL_k^p(y)\ o\ bien\ \ y \in HL_k^p(x)\ \}, \\ \infty si\ \ x\ \notin\ HL_k^p(y)\ o\ bien\ \ y\ \notin\ HL_k^p(x) \end{cases}$$

MATRICES ESTRUCTURALES SECUNDARIAS

El primer paso por tanto, será el de definir un concepto llamado derivación en el contexto de la extensión de horquilla. Decimos que la secuencia x deriva a la secuencia w, y lo denotamos como x → w, si y solo si $w \in HL_k(x)$, es decir, si w puede formarse a partir de x mediante la operación de extensión de horquilla. Si x es un segmento de w, $w \in HL_k(x)$, y w es de longitud n, definimos la máxima derivación de w a partir de x como la secuencia de un número p de pasos de derivación $w_0 \rightarrow w_1 \rightarrow \cdots \rightarrow w_p$, donde:

- $w_0 = x\ y\ w_p = w\ y\ w_i\ como\ pasos\ intermedios$
- Para cualquier i, con $p > i \geq 0$, las secuencias en cada paso de derivación están representadas o bien como $w_i = w[s_i..t_i]\ y\ w_{i+1} = w[s_i..t_{i+1}]$, donde t_{i+1} es el valor máximo de t tal que,$w[s_i..t] \in HLS_k(w[s_i..t_i])$, o bien como $w_i = w[s_i..t_i]\ y\ w_{i+1} = w[s_{i+1}..t_i]$, donde s_{i+1} es el valor mínimo de s tal que, $w[s..t_i] \in HLS_k(w[s_i..t_i])$. En otras palabras, estas ecuaciones definen cómo encontrar las elongaciones por un extremo y por otro producidas en cada paso de derivación.

En el caso de que p = 1, la derivación $x \rightarrow w$ sería la máxima derivación de w a partir de x. Vamos a asumir pues para continuar con la definición que p > 1. Esto equivale a establecer que para $p - 1 > i \geq 0$, en cada paso de derivación tendremos uno de los siguientes casos:

- $w_i \rightarrow w_{i+1}, w_i = w[s_i..t_i], w_{i+1} = w[s_{i+1}..t_{i+1}]\ y\ existe\ un\ valor\ t'_{i+1} > t_{i+1}\ tal\ que, w[s_i..t'_{i+1}] \in HLS_k(w_i)$
- $w_i \rightarrow w_{i+1}, w_i = w[s_i..t_i], w_{i+1} = w[s_{i+1}..t_{i+1}]\ y\ existe\ un\ valor\ s'_{i+1} < s_{i+1}\ tal\ que, w[s'_{i+1}..t_i] \in HLS_k(w_i)$

En otras palabras, sin perder la generalidad, se puede asegurar que en cada paso de derivación existirá un índice a un extremo u otro de la secuencia hasta el cual se producirá la elongación. La existencia de estos índices en cada iteración hará que la matriz se pueda ir recorriendo de manera sistemática con ayuda de las matrices estructurales de la primera parte del algoritmo y se pueda almacenar sus índices máximos y mínimos. Esta acción se realizará mediante las matrices estructurales Cp y Cs, que se definen de la siguiente forma:

- $Cs\ [i][j] =\ t, si\ y\ solo\ si\ w[i..t] \in HLSk\ (w[i..j])\ y\ w[i..t'] \notin HLS_k(w[i..j]) para\ todos\ los\ índices\ t'tales\ que\ n\ \geq t' > t$
- $Cp\ [i][j] =\ s, si\ y\ solo\ si\ w[s..j] \in HLPk\ (w[i..j])\ y\ w[s'..j\ ] \notin HLPk\ (w[i..j]) para\ todos\ los\ índices\ s'tales\ que\ \ 1 \leq s' < s$

No obstante, para calcular estas matrices son necesarias en primer lugar, otras que nos indiquen cuáles son tanto los máximos valores de t como los mínimos de s. Estas matrices serán P's y P'p, que se definen como:

$$P_s'[i][j] = \max\{t \mid j \leq t \leq n, P_s[j][t] = i\}$$
$$P_p'[i][j] = \min\{s \mid 1 \leq s \leq i, P_p[s][i] = j\}$$

En otras palabras, lo que buscaremos aquí es, con ayuda de las matrices estructurales Ps y Pp, las máximas elongaciones posibles de cada segmento de w tanto a un extremo como a otro. Nótese que si no se cumpliera la igualdad en las matrices estructurales Ps y Pp en una posición, la matriz en dicha posición tendría un valor de 0.

Una vez obtenidos estos valores máximos y mínimos, ya podemos inicializar las matrices Cs y Cp, que se llenarán de la siguiente forma:

$$C_s[i][j] = 0 \; for \; i > j \; or \; j = n, y \; C_s[i][j] = \max\{P_s'[i][j], C_s[i+1][j]\}$$
$$C_p[i][j] = 0 \; for \; i > j \; or \; i = 1, y \; C_p[i][j] = \max\{P_p'[i][j], C_p[i][j-1]\}$$

Los valores triviales siguen las mismas suposiciones que en las matrices Ps y Pp, ya que no existe elongación más alla del índice 1 o n, y el funcionamiento a la hora de calcular la matriz trabaja de la misma manera que la matriz M, es decir, se llenan primero las palabras más pequeñas y se va aumentando de tamaño, y de izquierda a derecha. En este caso, los valores de las matrices C corresponderán a los valores de cada paso de derivación posible. Se calcularán de manera sistemática y no será necesario recorrer las secuencias varias veces.

MATRIZ H CALCULADORA DE LA DISTANCIA

Finalmente, tras calcular las matrices estructurales Cs y Cp, podemos computar la distancia de extensión de horquilla definida por la longitud de horquilla *k* entre x y w. La estrategia que usaremos es una mezcla de programación dinámica y estrategia voraz (en inglés, *greedy strategy,* buscadora del óptimo). Analizaremos de nuevo en orden creciente de longitud (programación dinámica), todos los segmentos de w y construiremos para cada uno de ellos los segmentos de w que pueden ser derivados de ellos extendiéndolos tanto como sea posible usando la operación de extensión de horquilla, como hacíamos en cada paso de una derivación máxima de x a w, buscábamos la máxima elongación posible y la almacenábamos (programación optimizada). Al mismo tiempo, contaremos para cada una de las palabras construidas el mínimo número de pasos de derivación necesarios para obtener dicho segmento a partir de x (es decir, la máxima derivación de x → w[i..j]) , y almacenaremos estos valores en una matriz de n x n llamada matriz H. De esta forma, la matriz H[i][j] almacenará, al final de la computación, el valor de $HLD_k(x$, w[i..j]). Si sustituimos i y j por 1 y n (índices relacionados con la secuencia w completa) obtendremos la máxima derivación de x → w. No obstante, esta derivación máxima no puede calcularse directamente, sino que recurrimos, como hemos contado a la programación dinámica.

Por definición, inicialmente, establecemos un valor de ∞ a todos los índices i y j comprendidos entre 1 y n, y establecemos una H[i][j] = 0, si w[i..j] = x. Además, para un par de índices i y j, con $i < j$ y $j - i + 1 > 2k$, establecemos:

$$H[i]\big[Cs[i][j-k+1]\big] = \min\big\{H[i]\big[Cs[i][j-k+1]\big], 1 + H[i][j]\big\}, if \; Cs[i][j-k+1] \neq 0,$$
$$H\big[Cp[i+k-1][j]\big][j] = \min\big\{H\big[Cp[i+k-1][j]\big][j], 1 + H[i][j]\big\}, if \; Cp[i+k-1][j] \neq 0.$$

Por tanto, para cada segmento w[s..t] de w, verificamos todas las formas posibles en que este segmento puede ser derivado de otro segmento de la secuencia aplicando las reglas de máxima derivación de w a partir de x. Cuando estemos buscando una derivación $w[i..j] \rightarrow w[s..t]$, ya habremos computado H[i][j], y por lo tanto podremos actualizar si es necesario el valor de H[s][t]. Además, las relaciones anteriores llevan a la correcta computación de los valores de la matriz y a la distancia $HLD_k(x, w)$, que se almacenará en la última posición analizada, es decir, H[1][n].

A continuación, se muestra el algoritmo que calcula la matriz H en la figura 7.

Algorithm 3 $HLD_k(x, w)$: returns $HLD_k(x, w)$ $(2k + 1 < |x| < |w|)$

```
1: Initialize array H;
2: Compute matrices C_s and C_p;
3: for l = 2k to n do
4:    for i = 1 to n − l + 1 do
5:       j = i + l − 1;
6:       if C_s[i][j − k + 1] ≠ 0 then
7:          H[i][C_s[i][j − k + 1]] = min{H[i][C_s[i][j − k + 1]], 1 + H[i][j]}
8:       end if
9:       if C_p[i + k − 1][j] ≠ 0 then
10:         H[C_p[i + k − 1][j]][j] = min{H[C_p[i + k − 1][j]][j], 1 + H[i][j]}
11:      end if
12:   end for
13: end for
14: Return H[1][n]
```

Figura 7: Algoritmo calculador de la matriz H

MODO DE USO DE LA HERRAMIENTA

Para finalizar este apartado, se muestra el desarrollo de una prueba de ejecución de la herramienta en un IDLE Python 3.4 (64 bit), visualizable en la figura 8.

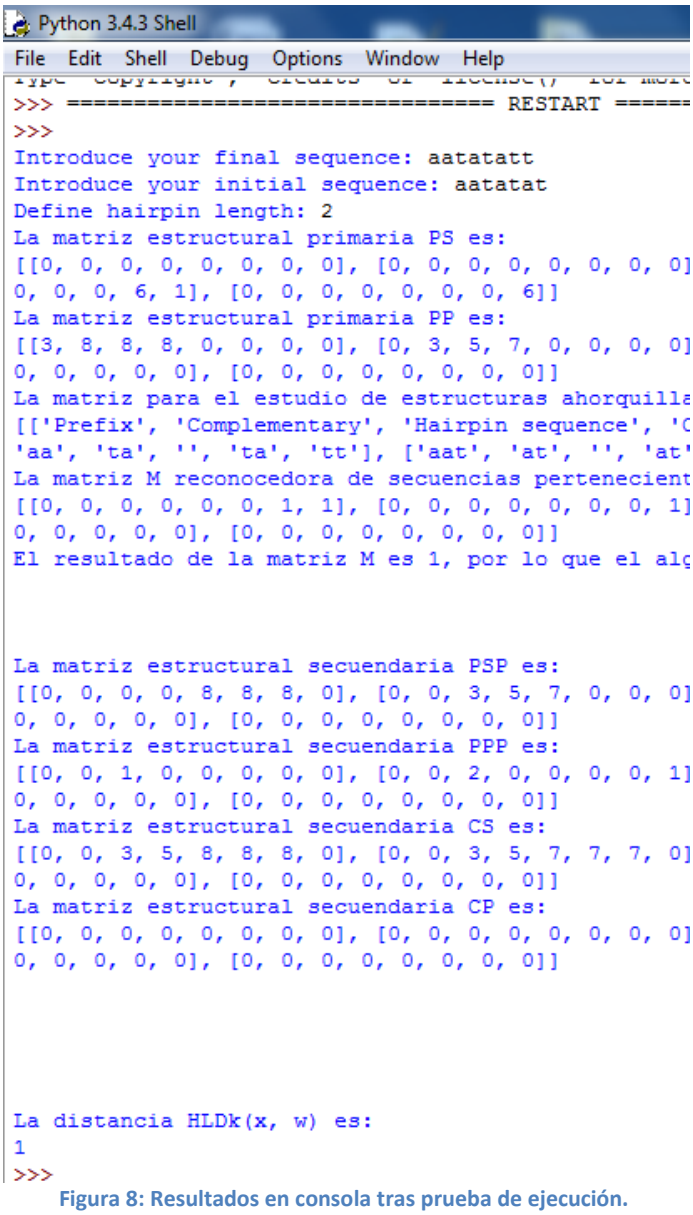

Figura 8: Resultados en consola tras prueba de ejecución.

Como se puede observar, la herramienta es de uso sencillo y los únicos parámetros que el usuario ha de introducir son los correspondientes a las dos secuencias sobre las que se calculará la operación de extensión de horquilla y la longitud de dicha horquilla.

RESULTADOS Y DISCUSIÓN

Una vez desarrolladas tanto la base matemática del algoritmo como su implementación, el siguiente paso será desarrollar un informe que muestre de una manera sencilla los resultados de nuestro analizador. Para esta tarea, se ha optado por escribir los resultados en un archivo de extensión .html, con la ayuda del módulo externo de Python HTML.py (accesible también bajo demanda junto con la herramienta), que formatea de una manera sencilla algoritmos realizados en python que tienen como resultados matrices, tablas o listas para devolverlas de una forma visual y entendible.

RESULTADOS DEL ANALIZADOR

Así, para una estructura dada, después del testado de la herramienta, los resultados obtenidos tienen la siguiente forma:

Hairpin Structures & Lengthening Results Report

The found hairpin structures among the sequence are:

Prefix	Complementary	Hairpin sequence	Complementary	Sufix
c	ggta	gattc	tacc	
cg	gtag	att	ctac	c
cgg	taga	t	tcta	cc

The results of the hairpin lengthening operation are:

Sequence X	Sequence W	Hairpin length	Hairpin distance (HLDk(x,w))
cggtagattctac	cggtagattctacc	4	1

Figura 9: Resultados obtenidos tras prueba de ejecución del analizador

Como podemos observar, los resultados tienen dos partes diferenciadas. En la primera, nuestro algoritmo devuelve una tabla que contiene las diferentes secuencias ahorquilladas que podemos obtener de nuestra secuencia W fragmentadas según el tipo de estructura. Las columnas “Complementary” muestran las secuencias de longitud k que pueden hibridarse y que definen la horquilla, la columna “Hairpin sequence” indica la secuencia en cadena simple formada por la hibridación de los fragmentos complementarios y las columnas “Prefix” y “Sufix” contienen las cadenas a ambos extremos de las cadenas complementarias. Cabe destacar que, como habíamos comentado en apartados anteriores, la herramienta encuentra horquillas simples, producidas tanto por hibridación de los extremos como por hibridación de secuencias intermedias. En la segunda parte del archivo obtenemos también una tabla con los datos introducidos para X, W y k y la distancia calculada para la distancia entre ambas secuencias.

USO DE BASES DE DATOS DE ESTRUCTURAS AHORQUILLADAS

Esta parte del proyecto se desarrolló, aunque no era el objetivo primordial, con el fin de desarrollar una aplicación práctica al analizador, que otorgara unos resultados útiles al usuario, más allá de la visualización de secuencias. De esta visualización se pueden obtener resultados

interesantes si se busca una secuencia en concreto o si se quiere ver si una pequeña región concreta pueda estar siendo modificada estructural o funcionalmente por una horquilla, pero esta utilidad no está contrastada y queda incompleta. En este sentido, cuando se estudian estructuras y motivos proteicos en una secuencia de nucleótidos, la base de datos más usada es la Protein Data Bank (PDB) (Sussman et al. 1998). Así, la búsqueda de identificadores de esta base de datos sí que podría dar al usuario una información más veraz. No obstante, esta búsqueda en la base de datos PDB no es directa en nuestro caso, sino que lo que nos interesa encontrar son secuencias que contengan complementariedades del tipo horquilla. Por lo tanto, en lugar de dirigirnos a la PDB nos deberíamos centrar en primer lugar en buscar estas estructuras ahorquilladas.

En nuestro caso, hemos optado por la *RNA CoSSMos Database* (Vanegas et al. 2012), una base de datos de caracterización de estructuras y motivos secundarios en secuencias de RNA. La búsqueda de una horquilla en esta base de datos se muestra en la figura 10.

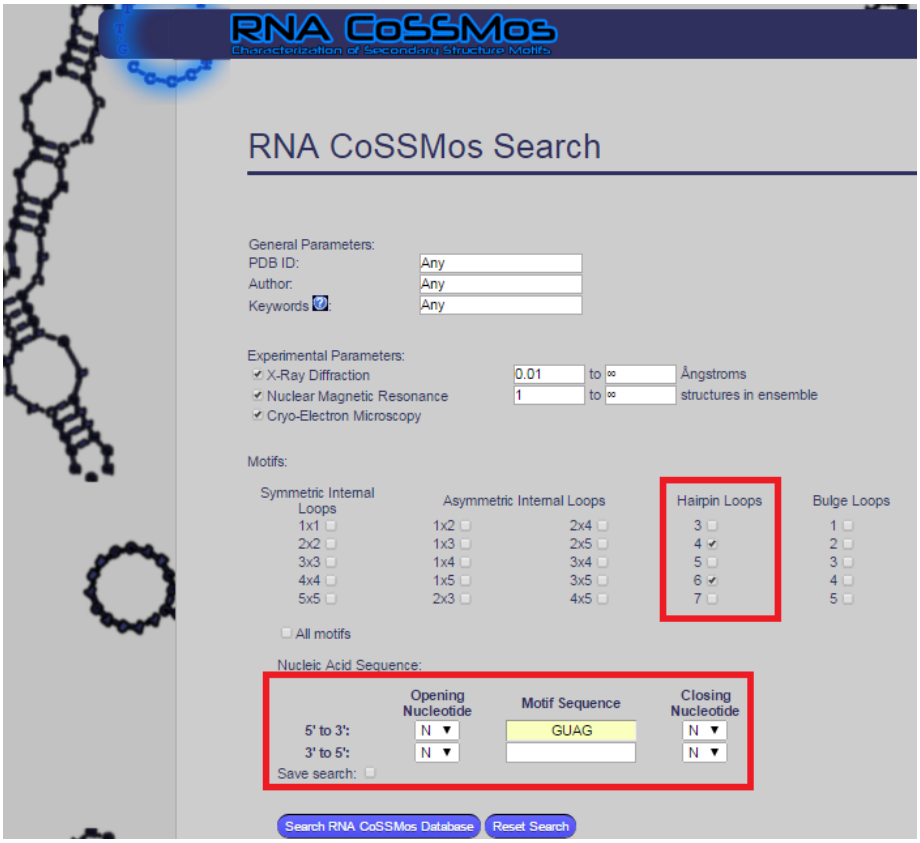

Figura 10: Página de búsqueda RNA CoSSMos Database

Como podemos observar en la figura, para conseguir una horquilla deberemos introducirla únicamente en el sentido 5'-> 3' y en formato RNA y también introduciremos la longitud de la horquilla. La base de datos nos calcula también todas las posibilidades para los nucleótidos de los extremos de la secuencia introducida. Una vez realizada la búsqueda, obtenemos los siguientes resultados mostrados en la figura 11.

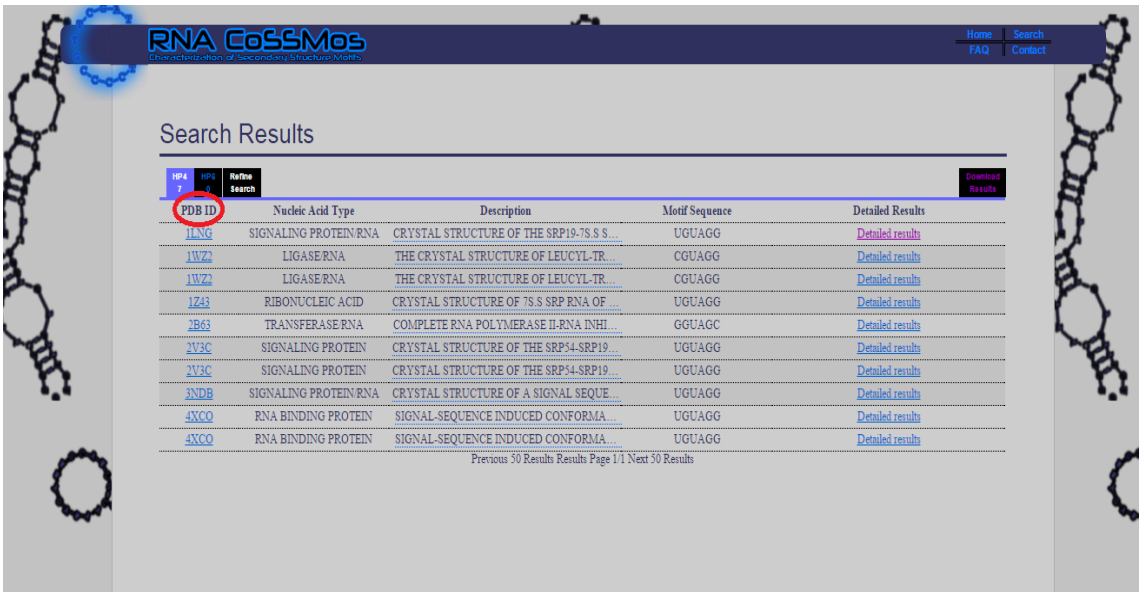

Figura11: Resultados de la búsqueda RNA CoSSMos Database

Las diferentes entradas para nuestra horquilla son listadas y descritas. Además, la base de datos nos devuelve un identificador en PDB para la secuencia en cuestión, que nos dirige a toda la información acerca de la secuencia, ilustrada en la figura 12.

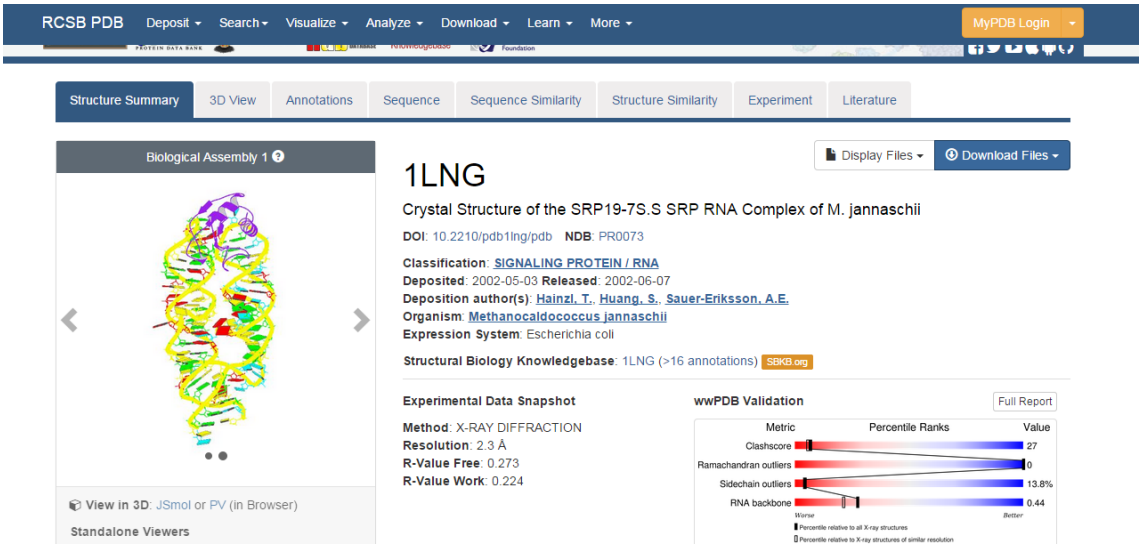

Figura 12: Representación de una página de búsqueda en Protein Data Bank (PDB)

Esta base de datos nos permite que nuestro analizador tenga una utilidad añadida que otorgue valor y sentido biológico a nuestros datos a través del contraste de información. Cabe destacar en este punto, que las bases de datos de estructuras ahorquilladas son, aunque existentes, relativamente escasas en entradas, y que no todas las estructuras ahorquilladas pueden encontrarse en ellas.

◘CONCLUSIÓN Y PERSPECTIVAS DE TRABAJO

Una vez obtenidos los resultados del analizador, podemos concluir que nuestra herramienta cumple con los objetivos del proyecto en tanto que nos otorga información sobre las estructuras ahorquilladas de nuestra secuencia y nos calcula el número mínimo de iteraciones de la operación de extensión de horquilla sobre X para obtener W, es decir, la distancia entre ambas secuencias. Además, gracias al sistema de computación, a la eficiencia de los algoritmos y a la base matemática en que se apoya la investigación, podemos concluir que el tiempo máximo del algoritmo es de $\mathcal{O}((|x|,|w|)^2)$ (Manea F. 2010).

Si bien en el proyecto definimos una potencial aplicación para nuestro analizador, el objetivo principal del proyecto es el diseño e implementación de la operación iterada de extensión de horquilla. De esta manera, después de la construcción de la herramienta, los retos futuros de la investigación de este tipo de operaciones en secuencias biológicas deberían consistir en la adaptación de la herramienta a técnicas que, aunque no siguen de manera exacta el procedimiento de nuestro analizador, sí que obtienen un resultado similar.

El primer ejemplo de este tipo de operaciones son las técnicas de amplificación de DNA que se basan en estructuras auto-horquilladas y la introducción de cebadores externos. En este sentido, aprovechando la eficiencia de nuestro algoritmo podrían introducirse estos cebadores y modelar (como paso previo a la experimentación) la estructura resultante mediante la elongación de los mismos. Los cebadores actuarían como secuencia complementaria y de esta manera podríamos hacer que la longitud de horquilla fuese diferente en cada iteración, lo cual supondría otra mejora y el aumento del número de resultados posibles. En segundo lugar, la herramienta podría tener una estrecha relación con la filogenia. En este sentido, el conocimiento de los procesos de formación y extensión de horquillas mediante modelos matemáticos sirve de base para un enfoque diferente a la hora de estudiar la evolución molecular, es decir, de entidades moleculares concretas. No obstante, para que esta opción resultara útil, no bastaría con el simple hecho de que la existencia de una horquilla produjera la evolución de la secuencia, sino que nuestra herramienta formaría parte de una herramienta más grande que tuviera en cuenta mutaciones, repeticiones en tándem y otros elementos que pudieran estar implicados en dicha evolución.

En resumen, nuestro analizador está preparado para otorgar una respuesta a la hipótesis de que una secuencia puede convertirse en otra a través de iterados pasos de extensión de horquilla, pero en este momento, los resultados no pueden llegar a considerarse explicativos, ya que deberían ser complementados con otro tipo de herramienta y comparados con algún modelo experimental específico de la operación que, a día de hoy, no existe.

BIBLIOGRAFÍA

- BARABAS, O.; RONNING, D.R; GUYNET, C.; HICKMAN, A.B.; TON-HOANG, B.; CHANDLER M.; DYDA, F. (2008) Mechanism of IS200/IS605 family DNA transposases: activation and transposon-directed target site selection. *Cell* 132:208-220.
- BOTTONI, P.; LABELLA, V.; MITRANA, V. (2006) Superposition based on Watson-Crick-like complementarity. *Theory of Computing Systems* 39(4):503-524.
- CHEPTEA, D.; MARTÍN-VIDE, C.; MITRANA, V. (2006) A new operation on words suggested by DNA biochemistry: hairpin completion, *Proc. Transgressive Computing*, 216-228.
- FAKRUDDIN, M.; MANNAN, K.S.B.; CHOWDHURY, A. (2013) Nucleic acid amplification: Alternative methods of polymerase chain reaction. *Journal of Pharmacy & Bioallied Sciences* 5(4):245-252.
- HOPCROFT, J. AND ULLMAN, J. (1979) *Introduction to autómata theory, Languages and computation*. Addison-Wesley, 554 pp.
- KHAN, S. A. (2005) Plasmid rolling-circle replication: highlights of two decades of research. *Plasmid* 53**:**126-136.
- KORNBERG, A. AND BAKER, T.A. (1992) *DNA replication*. W. H. Freeman & Co. New York. 931 pp.
- LASLETT, D. AND CANBACK, B. (2004) ARAGORN, a program to detect tRNA genes and tmRNA genes in nucleotide sequences. *Nucleic Acids Research* 32(1):11-16.
- MANEA F. (2010) A series of algorithmic results related to the iterated hairpin completion. *Theoret. Comput. Sci.,* 411 (48), pp. 4162–4178.
- MANEA F.; AND MITRANA, V. (2007) Hairpin completion versus hairpin reduction*, Computation in Europe CiE 2007*, LNCS 4497:532-541.
- MANEA, F.; MARTÍN-VIDE, C.; MITRANA, V. (2010) Hairpin lengthening, *Springer-Verlag* 296-306.
- MANEA, F.; MITRANA, V.; YOKOMORI, T. (2009) Two complementary operations inspired by the DNA hairpin formation: completion and reduction. *Theoret. Comput. Sci*., 410 (4–5), pp. 417–425.
- MARCO, A.; NINOVA, M.; RONSHAUGEN, M.; GRIFFITH-JONES, S. (2013) Clusters of microRNAs emerge by new hairpins in existing transcripts. *Nucleic Acids Research* 41(16):7745-7752.
- PROCTOR, J.R. AND MEYER, I.M. (2013) CoFold: an RNA secondary structure prediction method that takes co-transcriptional folding into account. *Nucleic Acids Research* 41(9):e102.
- PRUSS, G.J. AND DRLICA, K. (1989) DNA supercoiling and prokaryotic transcription. *Cell* 56**:**521-523.
- REIF, J.H. AND MAJMUDER, U. (2008) Isothermal reactivating Whiplash PCR for locally programable molecular computation. *DNA Computing: 14th International Meeting on DNA Computing 2008.*

- SALZMAN, L.A. AND FABISCH, P. (1979) Nucleotide sequence of the self-priming 3' terminus of the single-stranded DNA extracted from the parvovirus Kilham rat virus. *Journal of Virology*.;30(3):946-950.
- SUSSMAN, J.L.; LIN, D.; JIANG, J. (1998) Protein Data Bank (PDB): database of three-dimensional structural information of biological macromolecules. *Acta Crystallogr D Biol Crystallogr* 54(Pt 6 Pt 1):1078–1084.
- VANEGAS P.; HUDSON, A.; DAVIS, A.; KELLY, S.; KIRKPATRICK, C.; ZNOSKO, B. (2012) RNA CoSSMos: Characterization of Secondary Structure Motifs—a Searchable Database of Secondary Structure Motifs in RNA Three-Dimensional Structures. *Nucleic Acids Research* 40.Database issue D439–D444.
- WALLACE, MI.; YING, L.; BALASUBRAMANIAN, S.; KLENERMAN, D. (1992) Non-Arrhenius kinetics for the loop closure of a DNA hairpin. *Proceedings of the National Academy of Sciences of the United States of America*. 2001;98(10):5584-5589.