Aplicación bioinformática para predicción de genes regulados por microARNs en plantas

Tesina de grado presentada
por

Uciel Pablo Chorostecki
C-3915/2

al
Departamento de Ciencias de la Computación
en cumplimiento parcial de los requerimientos
para la obtención del grado de

Licenciado en Ciencias de la Computación

Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario
Av. Pellegrini 250, Rosario, Repúlica Argentina

Mayo 2011

Supervisores

Director
Dr. Javier Palatnik

Biología Molecular
IBR
Suipacha 531
Rosario, Argentina

Co-Director
Dra. Pilar Bulacio

Cifasis
27 de Febrero 210bis
Rosario, Argentina

Resumen

Los microARNs (o miARNs) son ARN no codificantes que regulan la expresión génica en animales y plantas, implicados en procesos biológicos muy variables, como el desarrollo, la diferenciación y el metabolismo. Estos pequeños ARNs de aproximadamente 21 nucleótidos reconocen secuencias parcialmente complementarias en los ARNm blanco, provocando su corte o arresto de la traducción.

Este trabajo propone estudiar en forma automatizada a los microARNs en plantas, su biogénesis y los genes que regulan, a través de un enfoque multidisciplinario. Para esto presentaremos una estrategia bioinformática para la identificación de genes blancos de microARNs y además una herramienta web para el ánalisis y selección de los mejores genes blancos candidatos. Considerando que muchos de estos ARNs pequeños están ampliamente distribuidos en plantas, la herramienta a desarrollar estará basada principalmente en la conservación durante la evolución de la interacción del par microARN-gen blanco en distintas especies.

Agradecimientos

El agradecimiento no es sólo por este trabajo ni las ayudas brindadas a nivel académico, sino que es a todos los que estuvieron conmigo en todos estos años de la carrera en cada momento acompañandome.

A mi familia en general por el apoyo. A mis viejos Ali y JK por enseñarme siempre con el ejemplo. A mis hermanos Iváncho y Nati que me bancaron a diario. A mis abuelas que confiaron en mí y estuvieron pendientes en cada materia rendida.

A mi nueva familia, mis cuñados y primos políticos.

A mis directores. Javier por darme esta gran oportunidad y Pilar por ayudarme con la escritura de este trabajo.

A mis amigos de la facultad. A Bibi y Berni que me acompañaron en casi todo el camino, porque juntarse a estudiar con ustedes es un lujo. A Damián, Julián, Germán y Juan que innumerables veces me dieron una mano. A los que me crucé en la última recta como Joaquín y mis amigos viajeros Checho Ezequiel y Zeta.

A mis compañeros y amigos del laboratorio. A Juan, Juli y Nico no sólo por las incontables ayudas sino también por todas las charlas (académicas o no) que tuvimos juntos. A Martín y Vale por las grandes ayudas en este trabajo. Al resto, Edgard, Sil, Carla, Rama y Ana por las críticas en los seminarios.

A mis amigas, las chicas del Poli, en especial a Flor que fue una de las "culpables" de mi destino y que además me ayudó a marcar este camino.

A los docentes, por transmitirnos esa pasión por lo que hacemos. Más allá del conocimiento a nivel académico, hacer que tengamos interés en lo que queremos ser.

A mis amigos de siempre, los del Poli, por dejarme compartir tantos años inolvidables con ustedes. Por dejarme ser parte.

Finalmente a Caro que sin ella todo esto (y lo que sigue) hubiera sido mucho más difícil. Por apoyarme siempre y por acompañarme sin importar la distancia. Simplemente, por estar conmigo.

A todos muchas gracias!

Índice general

Índice de figuras

Capítulo 1

Motivación y objetivos

1.1. Motivación

Los organismos contienen unidades de información llamadas genes. El número absoluto aproximando de genes en seres humanos es de 25.000 mientras que en una planta como *Arabidopsis thaliana* es de 30.000 y en una bacteria como *Escherichia coli* es de 3.000.

Lo que hace a la complejidad de los seres vivos no radica en el número de genes, sino en cómo parte de estos genes son utilizados. En determinadas células específicas están activos (o expresados) solamente ciertos genes. De la misma manera cuando hay un cambio en el ambiente se inactivan (o se apagan) ciertos genes y se activan otros. La combinación de genes activos e inactivos determina la respuesta y el comportamiento de la célula. Una de las grandes preguntas de la biología es de qué manera se coordinan dinámicamente estas redes de genes para formar un organismo complejo funcional.

En los últimos diez años se ha demostrado que sistemas regulatorios basados en ARN pequeño son responsables en parte de esa regulación de la expresión génica. Los microARNs son unos de los mecanismos más importantes para regular la expresión génica y cuasan el silenciamiento, es decir la inactivación de genes. En seres humanos alrededor del 30 % de los genes están regulados por microARNs. Otra particularidad que tienen los microARNs es que pueden regular cuantitativamente el nivel de expresión de los genes, es decir que no necesariamente aparecen como una variable binaria sino más bien como una variable continua.

Los microARNs son esenciales a la hora de formar un organismo complejo, si no hay microARNs no se puede formar una planta, un animal o un ser vivo cualquiera. En los últimos años el avance tecnológico (basado en técnicas de secuenciación de segunda generación) ha permitido identificar la gran mayoría de microARNs. Es por esto que surge la siguiente pregunta. ¿Cuáles son los genes regulados por estos microARNs? Asimismo, debemos notar que existe una metodología capaz de reconocer a los microARNs y se han usado en distintos trabajos para la identificación de un gran número de microARNs en plantas [7].

En cambio los genes blancos que son regulados por estos microARNs no son triviales

de identificar directamente. En trabajos anteriores se han identificado genes blancos regulados por microARNs mediante distintos enfoques [27, 31, 8] pero esta predicción de genes blancos no es fácil, por lo tanto se necesita de una estrategia bioinformática con análisis estadísticos para poder predecir nuevos genes blancos.

Esta tesina busca responder este interrogante, es decir, cuáles son los genes blancos regulados por microARNs. Para encontrar respuestas, exploramos estrategias bioinformáticas y análisis estadísticos para predecir nuevos genes blancos (o targets) regulados por microARNs conservados en plantas. Cabe destacar que los microARNs cumplen funciones importantes en las plantas y regulan el tamaño y forma de la misma, como la respuesta a situaciones ambientales desfavorables. Por lo tanto, identificar los genes blancos de microARNs puede tener aplicaciones tecnológicas en el campo de la biotecnología.

1.2. Objetivos

Como objetivo general, este proyecto pretende estudiar en forma sistematizada a los microARNs en plantas, su biogénesis y los genes regulados por éstos. Con este fin diseñamos una estrategia y desarrollamos una herramienta bioinformática para facilitar la identificación de genes blancos de microARNs. Primeramente planteamos una estrategia informática de búsqueda de genes blancos, mediante la adaptación de una herramienta disponible e integración de nuevos módulos para expandir el alcance de dicha herramienta. A continuación realizamos un análisis estadístico de los resultados obtenidos al realizar búsquedas de genes blancos para microARNs conservados. Y por último diseñamos y desarrollamos un nuevo software a partir de datos almacenados en una base de datos para que colegas del laboratorio con formación netamente biológica lo utilicen como herramienta de estudio.

1.3. Aspectos considerados

Hay dos aspectos relevantes en este trabajo. Por un lado el aspecto informático asociado a la complejidad que presentan las búsquedas de genes blancos ya que existen distintas interacciones del par microARN-gen blanco posibles. Por otro lado, consideramos el aspecto biológico ya que luego de una predicción informática necesitamos hacer un análisis más detallado de los resultados para verificar que éstos tengan coherencia y relevancia biológica.

1.3.1. Aspecto Informático

El microARN interactúa con la secuencia blanco por complementariedad de bases. Como la interacción no es perfecta existen muchas interacciones posibles lógicas. Aunque se conocen empíricamente algunas interacciones no permitidas, el número de interacciones permitidas sigue siendo muy grande. Es por esto que los algoritmos que tienen en cuenta las interacciones imperfectas demandan mucho tiempo computacional.

1.3.2. Aspecto Biológico

En el aspecto biológico surge otro problema. Computacionalmente se puede predecir una interacción permitida que fue utilizada para identificar genes blancos anteriormente. Pero luego, para encontrar nuevos genes blancos esta interacción puede no ser biológicamente válida. Puede haber varias explicaciones a que un microARN sea computacionalmente predicho que pueda regular varios genes, en realidad no lo haga *in vivo*. Una explicación posible es que el microARN se exprese en células de raíz y el gen blanco se exprese en hoja. Entonces es imposible que dicho gen blanco sea regulado por ese microARN al no estar expresándose ambos en las mismas células.

En general las secuencias de nucleótidos pueden sufrir mutaciones (alteraciones en la información genética) a no ser que exista presión de selección para que esto no ocurra. Si la regulación de un gen por un microARN es importante biologicamente, entonces el sitio reconocido por el microARN debería estar conservado en distintas especies de plantas donde el microARN también se encuentra conservado.

1.4. Organización de la tesina

En el capítulo 2, para que este trabajo sea autocontenido, damos una breve y simple explicación de los principios básicos de Biología molecular. A continuación en el capítulo 3 describimos las herramientas que adpatamos, los nuevos módulos y las bases de datos utilizadas. Una vez mostrada las herramientas, en el capítulo 4, detallamos distintos enfoques que utilizamos para validar nuestra estrategia de búsqueda de genes blancos. A continuación, en el capítulo 5, mostramos los resultados obtenidos a partir de la estrategia y la comparación de genes blancos para microARNs y para sus secuencias *scramble* y también los distintos controles utilizados para validar la estrategia. En el capítulo 6 describimos una herramienta final desarrollada, en la cuál biólogos moleculares se basaron para encontrar nuevos genes blancos para microARNs conservados. Y por último en el capítulo 7 daremos una breve conclusión de este trabajo y los distintos estudios relacionados que se pueden dar a partir de éste.

Capítulo 2

Introducción a la Biología

El objetivo de este capítulo es brindar al lector conocimiento básico de Biología relativos a este trabajo. Es por esto que, para que este trabajo sea autocontenido, se da una breve y simple explicación de los principios básicos de Biología molecular. Comenzaremos con un concepto importante llamado Dogma Central de la Biología [13]. Luego describimos los elementos que constituyen a este concepto que son el ADN, el ARN y la Proteína. A continuación detallamos los procesos secuenciales que constituyen el Dogma Central, que establece que la información fluye desde el ADN al ARN y de éste último a las proteínas. Y por último damos una idea aproximada de qué es un microARN y su relevancia en la comunidad científica.

2.1. Dogma Central de la Biología

Las células realizan un proceso en dos pasos llamados transcripción y traducción para leer cada gen y producir la cadena de aminoácidos que forman una proteína. Este flujo de información constituye el Dogma Central de la Biología, tal como está descrito en la figura 2.1

.

Figura 2.1: Dogma Central de la Biología

2.1.1. ADN, ARN y proteína

Para comprender al Dogma Central describiremos a los elementos que lo constituyen. Estos son el ADN, el ARN, el ARN mensajero (ARNm), la proteína. Asimismo veremos unos conceptos relacionados con el código genético para comprender la relación entre el alfabeto del ADN y el ARN con respecto al alfabeto de la proteína.

ADN

El ADN es el portador de la información genética en las células. La estructura del ADN es una doble hélice, muy similar a una escalera de caracol formando una espiral como se muestra en la figura 2.2. Las bases del ADN se encuentran en pares, los cuales hacen los escalones de la escalera. Los laterales de la escalera son la médula estructural del ADN. Estos laterales no contienen información, sólo sostienen a las bases en su posición correcta.

Figura 2.2: Estructura del ADN

Las bases del ADN son la citosina (C), la timina (T), la adenina (A) y la guanina (G) que junto a un grupo fosfato y la desoxirribosa forman un nucleótido. Lo que distingue a cada nucleótido de otro es la base nitrogenada. La disposición secuencial de estas cuatro bases a lo largo de la cadena es la que codifica la información genética, por ejemplo, una secuencia de ADN puede ser ATGCTAGATCGC... En el ADN se da un apareamiento complementario de las bases donde se aparean G con C y A con T.

El ADN que constituye el genoma de un ser vivo puede ser subdividido en partes de información llamados genes. Cada gen contiene información para la producción de una proteína única, la cual realizará una función especializada en la célula. En particular el genoma humano contiene más de 25.000 genes.

ARN y ARNm

Al igual que el ADN, el ARN contiene 4 bases nucleotídicas, pero en el ARN la base uracilo (U) reemplaza a la timina (T). El ARN es también el material genético de muchos virus.

El ARN mensajero (o ARNm) lleva la información del ADN a los ribosomas, el lugar de la síntesis de proteínas. La secuencia de nucleótidos del ARNm determina la secuencia de aminoácidos de la proteína. Por ello, el ARNm es denominado ARN codificante. El ARNm sigue esencialmente las mismas reglas que el ADN para formar los pares de bases: G forma un par con C y A forma otro par con U, aunque en general, el ARN es una cadena única en vez de una doble helice como el ADN. No obstante, muchos ARN no codifican proteínas y reciben el nombre de ARN no codificantes.

Proteína

Las proteínas son biomoléculas formadas por cadenas lineales de aminoácidos. Las proteínas desempeñan un papel fundamental para la vida y son las biomoléculas más versátiles y más diversas. Son imprescindibles para el organismo. Realizan una enorme cantidad de funciones diferentes de tipo estructural, reguladora, transportadora e inmunológica entre otras. Las proteínas se sintetizan dependiendo de cómo se encuentren regulados los genes que las codifican. Por lo tanto, son susceptibles a señales o factores externos.

Código Genético

El código genético es el conjunto de normas por las que la información codificada en el material genético (secuencias de ADN o ARN) se traduce en proteínas (secuencias de aminoácidos) en las células vivas. El código define la relación entre secuencias de tres nucleótidos, llamadas codones, y aminoácidos. Un codón se corresponde con un aminoácido específico.

La secuencia del material genético se compone de cuatro bases nitrogenadas distintas, que tienen una función equivalente a letras en el código genético: adenina (A), timina (T), guanina (G) y citosina (C) en el ADN y adenina (A), uracilo (U), guanina (G) y citosina (C) en el ARN. Debido a esto, el número de codones posibles es 64, de los cuales 61 codifican aminoácidos (siendo además uno de ellos el codón de inicio, AUG) y los tres restantes son sitios de parada (UAA, UAG y UGA). La secuencia de codones determina la secuencia aminoacídica de una proteína en concreto, que tendrá una estructura y una función específica. En la figura 2.3 se muestra la tabla del código genético que indica qué codones codifican cada uno de los aminoácidos, sin incluir el codón de comienzo ni los de parada.

2.1.2. Procesos involucrados

Una vez definidos los elementos que constituyen al Dogma Central de la Biología podemos describir los procesos que están involucrados en éste .

Replicación

La replicación es el mecanismo que permite al ADN duplicarse (es decir, sintetizar una copia idéntica). De esta manera, de una molécula de ADN única se obtienen dos

Aminoácidos	Codones
Ala (A)	GCU, GCC, GCA, GCG
Arg (R)	CGU, CGC, CGA, CGG, AGA, AGG
Asn (N)	AAU, AAC
Asp (D)	GAU, GAC
Cys (C)	UGU, UGC
Gln (Q)	CAA, CAG
Glu (E)	GAA, GAG
Gly (G)	GGU, GGC, GGA, GGG
His (H)	CAU, CAC
Ile (I)	AUU, AUC, AUA
Leu (L)	UUA, UUG, CUU, CUC, CUA, CUG
Lys (K)	AAA, AAG
Met (M)	AUG
Phe (F)	UUU, UUC
Pro (P)	CCU, CCC, CCA, CCG
Sec (U)	UGA
Ser (S)	UCU, UCC, UCA, UCG, AGU, AGC
Thr (T)	ACU, ACC, ACA, ACG
Trp (W)	UGG
Tyr (Y)	UAU, UAC
Val (V)	GUU, GUC, GUA, GUG

Figura 2.3: Código genético

o más "clones" de la primera. Esta duplicación del material genético se produce de acuerdo con un mecanismo semiconservador. Lo que indica que las dos cadenas complementarias del ADN original, al separarse, sirven de molde cada una para la síntesis de una nueva cadena complementaria de la cadena molde. De esta forma, cada nueva doble hélice contiene una de las cadenas del ADN original. Gracias a la complementariedad entre las bases que forman la secuencia de cada una de las cadenas, el ADN tiene la importante propiedad de reproducirse idénticamente, lo que permite que la información genética se transmita de una célula madre a las células hijas y sea la base de la herencia del material genético.

Transcripción

La transcripción es el primer proceso de la expresión génica. Implica copiar la secuencia de ADN en la forma de ARN mensajero (ARNm). La molécula de ARNm transporta la información para hacer una proteína desde el núcleo de la célula (donde se encuentra el ADN) hacia el citoplasma (donde se ubica la maquinaria para hacer proteínas).

Traducción

El segundo paso de la síntesis proteica (parte del proceso general de la expresión génica) es llamado traducción. En el citoplasma, la información en el ARNm se traduce

por la maquinaria celular productora de proteínas, llamada ribosoma, la cual ensambla las proteínas.

Los ribosomas usan Código Genético para determinar la secuencia de aminoácidos codificada por el ARNm. Solamente la información contenida entre las señales de inicio (AUG) y terminación (UAA, UAG o UGA) de una molécula de ARNm es usada para producir una secuencia de aminoácidos. Después de la señal de inicio (AUG), el ribosoma lee tres nucleótidos a la vez. Cada grupo de tres nucleótidos, o codón, especifica un aminoácido en particular.

Para resumir estos conceptos y a modo de ilutración, mostramos en la figura 2.4 una secuencia de ADN de un gen que primeramente se transcribe y luego se traduce a proteína.

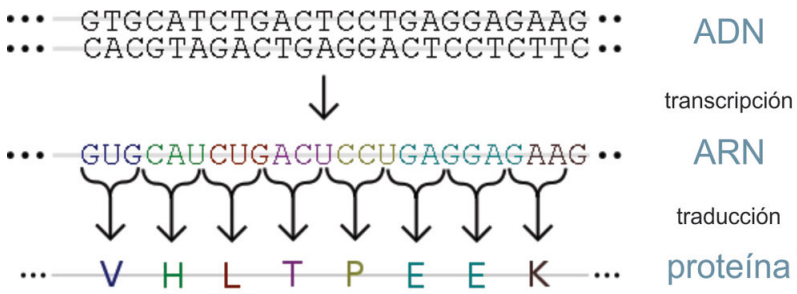

Figura 2.4: La secuencia de ADN de un gen que codifica para la secuencia de aminoácidos de una proteína.

2.2. MicroARNs

Los microARNs (miARNs) son ARN no codificantes que regulan la expresión génica en animales y plantas, implicados en procesos biológicos muy variables, como el desarrollo, la diferenciación y el metabolismo [4]. Estos pequeños ARNs de aproximadamente 21 nucleótidos reconocen secuencias parcialmente complementarias en los ARNm blanco, provocando su corte o inhibiendo la traducción.

Las secuencias de los microARNs conocidas se depositan en una base de datos que nuclea la información[1]. Cada microARN que se descubre recibe un número que lo identifica, como miR319 o miR396 que son microARNs de plantas. A su vez puede haber varios microARNs codificados por genes diferentes pero que tengan las misma o muy similar secuencia. Estos últimos se los puede distinguir con una letra, asi por ejemplo el miR396a y el miR396b difieren en el nucleótido número 21.

[1] http://mirbase.org/

2.2.1. Importancia de los microARNs

Los microARNs han saltado rápido a la primera plana del interés de la comunidad científica como un nuevo nivel en el control de la expresión génica en eucariotas. Primeramente se pensó que eran una excentricidad evolutiva únicamente encontrada en gusanos. La situación cambió radicalmente cuando en el año 2000 se reconoce que los microARNs se encuentran ampliamente difundidos en animales y plantas. La importancia de los microARNs en los seres multicelulares está claramente establecida, ya que la inactivación de DICER (enzima que genera los microARNs) tanto en ratones, en peces como en *Arabidopsis thaliana* es letal para el organismo [1, 33]. Mutantes parciales en DICER y otros genes que participan en el metabolismo de los microARNs presentan fenotipos pleiotrópicos (múltiples defectos) indicando la participación de los microARNs en múltiples procesos biológicos. Los cálculos actuales consideran que entre el 20 % y el 40 % de los genes de humanos se encuentran regulados por microARNs [3].

Si bien ha habido un gran avance en la identificación de microARNs, primero por clonado, luego por predicción bioinformática y finalmente por secuenciacion de segunda generacion, aún existen muchos interrogantes en este campo que avanza rápidamente.

Relevancia del problema

Los microARNs han sido descubiertos como un mecanismo general por el cual los organismos pluricelulares pueden regular la expresión génica. La mayor parte de los estudios comenzaron hace menos de diez años utilizando una combinación de técnicas de biología molecular, genómica y bioinformática. Los primeros esfuerzos se basaron en la construcción de bibliotecas de ARN pequeños y en identificarlos por clonado [7]. Una vez conseguido un número razonable de microARNs, se diseñaron algoritmos bioinformáticos para identificarlos en los genomas secuenciados [27].

Estudios recientes han puesto de manifiesto que los microARNs están estrechamente involucrados en distintas enfermedades de importancia. Algunos tienen relación con distintos tipos de Cáncer [11] y otros están relacionados con enfermedades cardíacas donde los niveles de expresión de microARNs específicos cambian en el corazón humano cuando están presentes dichas enfermedades [12].

El conocimiento en plantas ha avanzado muchas veces más rápidamente que en animales, en parte debido a que la alta homología entre el microARN y el sitio blanco en plantas hace más fácil la predicción de ARNm regulados por microARNs. Los resultados obtenidos en la pequña planta modelo de laboratorio: *Arabidopsis thaliana* han sido numerosamente extrapolados con éxito a los sistemas animales. Esto ha causado que varios de los laboratorios que tradicionalmente solo trabajaron en estos sistemas han abierto líneas en *Arabidopsis*.

Es interesante notar que el microARN humano miR196 regula la expresión de genes homeóticos guiando los ARNm a su degradación en un mecanismo similar al que se describe en plantas [9]. Además de estas aplicaciones básicas y en biomedicina, la manipulación de los niveles de microARNs propone importantes aplicaciones biotecnológicas en plantas. Al estar los microARNs utilizados en este proyecto conservados

en especies de interés agronómico, las aplicaciones potenciales de este trabajo podrían ser inmediatas.

Para ver graficamente la importancia de los microARNs en plantas mostramos dos enfoques distintos reflejados en las figuras 2.5 y 2.6. En la figura 2.5 tenemos distintas plantas transgénicas que sobreexpresan un microARN en particular. En este enfoque se regula negativamente todos los genes blancos por medio del microARN sobreexpresado. En esta figura mostramos en el lado izquierdo la planta silvestre o *wildtype* (es el fenotipo de la forma típica de una especie, como ocurre en la naturaleza) y en el lado derecho la planta sobreexpresante de un microARN. Por ejemplo en la figura 2.5 (d) la sobreexpresión del miR319 produce la degradación practicamente total de 5 factores de transcripción de la familia TCP y produce una planta donde se ve claramente afectada la morfología de las hojas [4].

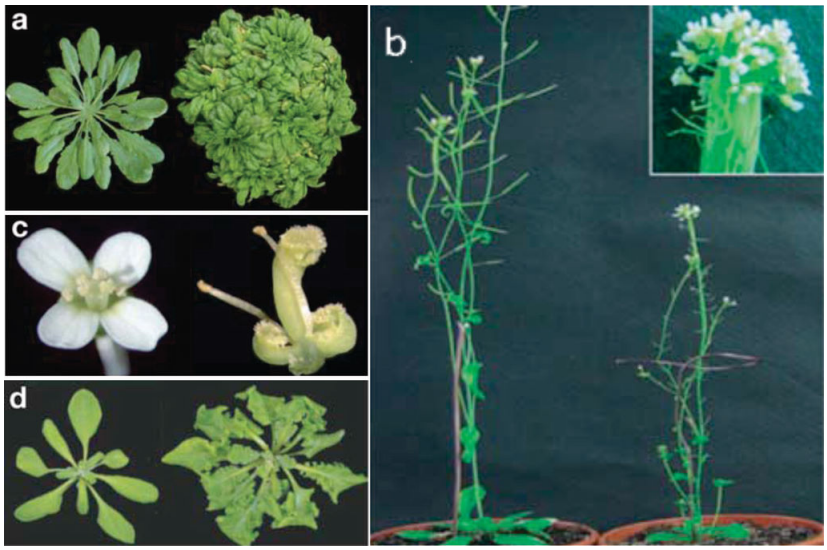

Figura 2.5: Ejemplos de fenotipos como resultado de sobreexpresar un microARN en *Arabidopsis*. Cada panel representa la planta silvestre (izquierda) comparado con un espécimen de una planta que sobreexpresa un microARN. (a) Las plantas que sobreexpresan el miR156 tienen mayor apertura en la hoja y disminución de la dominancia apical. (b) Las plantas que sobreexpresan el miR166 disminuyen en estatura y fertilidad y se amplían los tallos (recuadro). (c) Los órganos del verticilo floral de la sobreexpresante del miR172 son transformados en tejido de carpelo en vez de tener 4 sépalos y 4 pétalos. (d) El fenotipo mutante *jaw*-D resulta de la sobreexpresión del miR319, donde se ve afectada la morfología de la hoja.

Por otra parte en la figura 2.6 podemos ver distintas plantas transgénicas que expresan una versión de un microARN resistente, donde se introducen mutaciones silenciosas en el sitio complementario del microARN que alteran la regulación mediada por el microARN sin alterar el producto de la proteína modificada. De nuevo, mostramos en el

lado izquierdo la planta silvestre y en el lado derecho la planta transgénica. Por ejemplo en la figura 2.5 (a) las plantas que expresan el miR159 resistente MYB33 son reducidas en estatura y las hojas son rizadas hacia arriba.

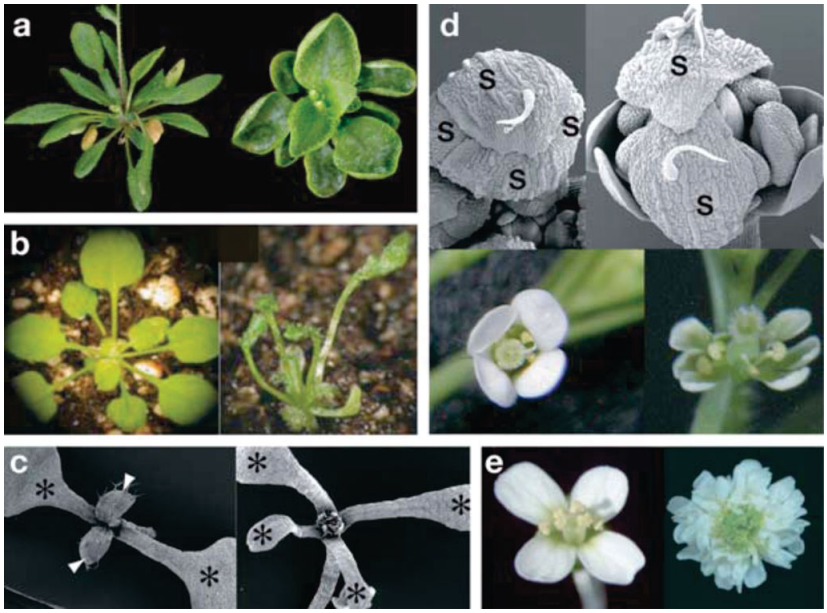

Figura 2.6: Fenotipos de transgénicas de Arabidopsis expresando un microARN resistente. Cada panel representa la planta silvestre (izquierda) comparado con una plata que expresa un microARN resistente. (a) Plantas que expresan el miR159 resistente son reducidas en estatura y las hojas rizadas hacia arriba. (b) Plantas que expresan el miR166 resistente tienen hojas reducidas con características adaxial en todo el perímetro de la hoja. (c) Las Plántulas con niveles normal del miR160 tienen 2 cotiledones (asteríscos) y 2 nuevas hojas (flechas) mientras que las transgénicas que expresan el miR160 resistente tienen hasta 4 cotiledones y una hoja emergente entre cada para de cotiledones. (d) Las flores con niveles normal del miR164 tienen 4 sépalos (arriba) y 4 pétalos (abajo), mientras que las transgénicas que expresan el miR164 resistente tienen 2 sépalos (arriba) y 6 pétalos (abajo). (e) Las flores que expresan el miR172 resistente tienen un número variable de los órganos florales, esta flor tiene pétalos numerosos y carece de verticilos internos.

Teniendo en cuenta estos dos enfoques mostramos que los microARNs son reguladores claves en muchas facetas del desarrollo de una planta, en este caso de *Arabidopsis*. Como los microARNs tenidos en cuenta en este trabajo están conservados en plantas, esto se puede extrapolar a otras plantas de interés comercial.

Capítulo 3

PatmatchMicro

Debido a la complejidad del problema adoptamos una arquitectura modular formado por distintos módulos, herramientas y base de datos. La misma se muestra en la figura 3.1.

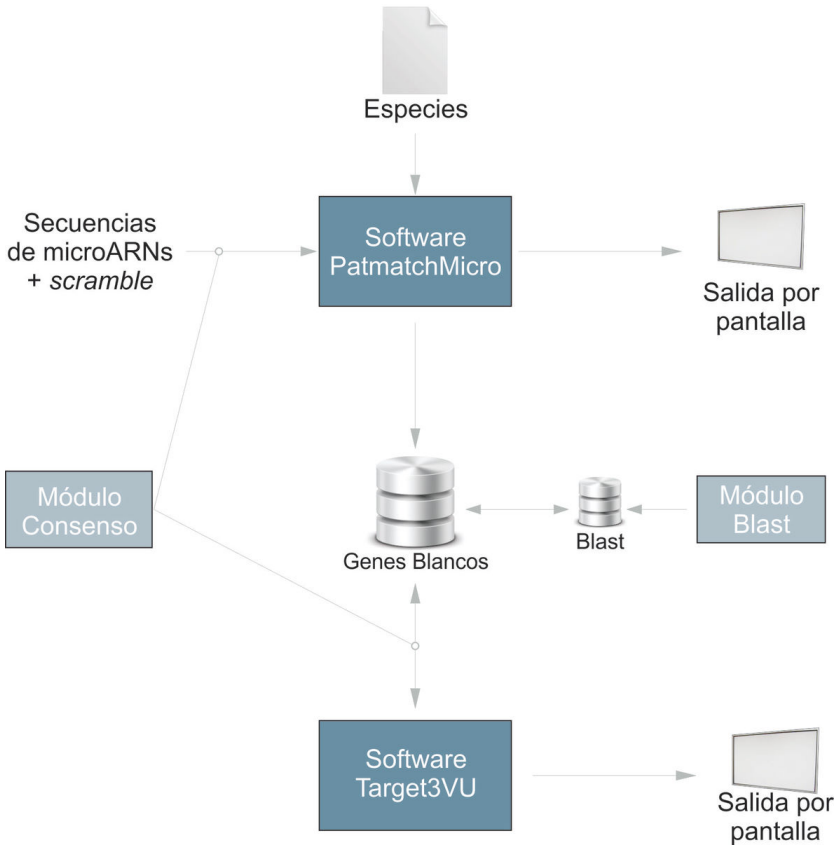

Figura 3.1: Arquitectura del trabajo. Donde incluye las herramientas adaptadas, módulos implementados y base de datos utilizadas.

Para alcanzar el objetivo de búsqueda de genes blancos, podemos considerar dos

etapas imporantes teniendo en cuenta esta arquitectura.

La primera etapa realiza el emparejamiento entre una secuencia de entrada y sus genes blancos para múltiples especies de plantas. Unas de las secuencias de entradas que se utilzan son los 22 microARNs obtenidos a partir de una secuencia consenso de 18 nucleótidos (ver detalles en la sección 3.3.1). A partir de ese análisis se obtienen fragmentos de secuencias que dan lugar a los denominados genes blancos. Estos genes blancos son guardados en una Base de datos, junto a otros datos útiles, para ser utilizados en la segunda etapa. Esta primera búsqueda se realizará con una herramienta denominada PatmatchMicro.

La segunda etapa usará los 22 microARNs conservados más la Base de datos de genes blancos obtenidas de la primera etapa. Esta información será la base de una herramienta final denominada Target3VU de donde se hará la selección de los mejores genes blancos candidatos para luego realizar la validación experimental. Esta herramienta que surge en una segunda etapa se mostrará en detalles en la sección 6.1.

Comenzaremos entonces a describir la primera etapa. Es por eso que detallamos el elemento de software PatmatchMicro. Esta herramienta que desarrollamos da a lugar a partir de una herramienta libre disponible existente denominada Patmach y unos módulos de software que son utilizados para darle más especificidad a la herramienta emergente. PatMatch (la herramienta que adaptamos) utiliza una herramienta denominada Nrgrep. La descomposición de PatmatchMicro en módulos se detalla en la figura 3.2.

En las próximas secciones vamos a describir los elementos que componen la herramienta PatmatchMicro, describiendo primero los compuestos más simples hasta la herramienta final. Por esto, describiremos primero la herramienta fundamental en este trabajo denominada Nrgrep, así como los distintos algoritmos que utiliza. A continuación describiremos la herramienta PatMatch, que utiliza Nrgrep como motor de búsqueda. Y por último describiremos los nuevos módulos desarrollados integrados junto a la herramienta PatMatch para terminar con la herramienta de la primera etapa de este trabajo (mencionada anteriormente) denominada PatmatchMicro.

3.1. Nrgrep

Nrgrep es una herramienta de búsqueda de patrones complejos. Está basado en un concepto único y uniforme: la simulación de paralelismo de bits mediante un autómata sufijo no determinista. Nrgrep está construido sobre un algoritmo llamado NBDM y permite tres diferentes tipos de patrones. Estos son patrones simples, patrones extendidos y expresiones regulares. A continuación vamos a dar una descripción de los algoritmos que Nrgrep utilza, así como la utilización del programa en sí. Para profundizar aún más sobre Nrgrep consultar [17].

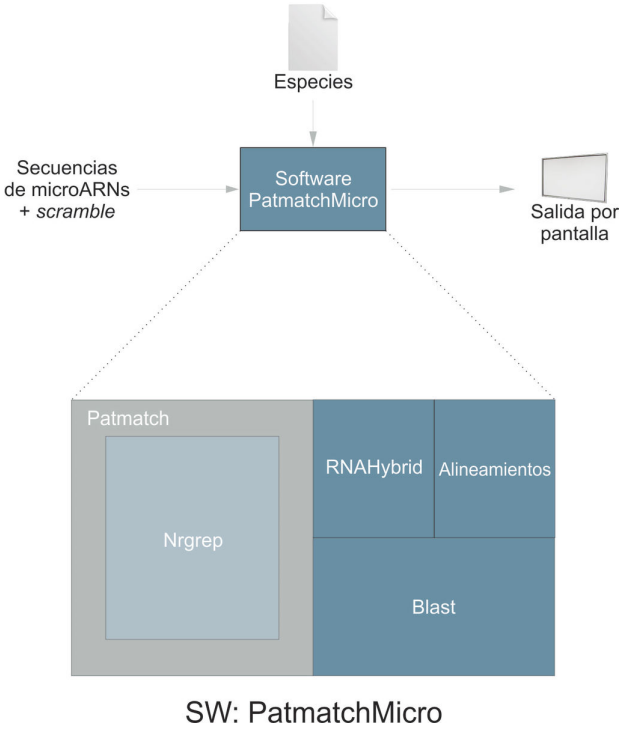

Figura 3.2: Primera etapa de la estrategia y detalle de los módulos del software: PatmatchMicro

3.1.1. Notación

Consideramos el texto como una secuencia de n caracteres, $T = t_1 \cdots t_n$ donde $t_i \in \Sigma$. Σ es el alfabeto del texto y su tamaño se denota con $|\Sigma| = \sigma$. Entonces tenemos tres tipos de patrones distintos:

- Patrones simples:
  Un patrón simple es una secuencia de caracteres o clase de caracteres. Sea m el número de elementos en la secuencia, entonces un patrón simple P se escribe como $P = p_1 \cdots p_m$ donde $p_i \subseteq \Sigma$. Decimos que P coincide *(matches)* en la posición del texto *i+1* cuando $t_i \in p_i$ para $i \in 1 \ldots m$. Los patrones simples se escriben directamente, mientras que otra clase de caracteres se escriben entre corchetes. El primer caractér dentro de corchetes puede ser "^" que refiere a que la clase es exactamente el complemento de lo que se especifico. El resto es una simple enumeración de caracteres de las clases, salvo que se permiten rangos. "xy" significa todos los caracteres entre x e y incluidos. Finalmente el caractér "." representa una clase igual al alfabeto completo y el caractér "#" representa la clase de todos los separadores.

- Patrones extendidos:
  Las operaciones que se permiten son: especificar clases (o caracteres) opcionales

y permitir la repetición de una clase o caractér. La notación que se usa para agregar un símbolo después del caractér afectado son: "?" que significa una clase opcional, "*" que significa que la clase puede aparecer cero o más veces y "+" que puede aparecer una o más veces.

- Expresiones regulares:
  Una expresión regular es el patrón mas complejo que se permite. Se define de la siguiente manera:

  i) elementos básicos: cualquier caractér y la cadena vacía ε son expresiones regulares.

  ii) paréntesis: si e es una expresión regular entonces lo es (e), que coincide con la misma cadena. Se usa para cambiar la precedencia.

  iii) concatenación: Si e_1 y e_2 son expresiones regulares, entonces $e_1 \,.\, e_2$ es una expresión regular que coincide con una cadena x, si y sólo si, x se puede escribir como $x = yz$. Donde e_1 coincide con y y e_2 coincide con z.

  iv) unión: Si e_1 y e_2 son expresiones regulares, entonces $e_1 \mid e_2$ es una expresión regular que coincide con una cadena x, si y sólo si e_1 o e_2 coinciden x.

  v) clausura de kleen: si e es una expresión regular, entonces, $e*$ es una expresión regular que coincide con una cadena x, si y sólo si, para algún n, x se puede escribir como $x = x_1 \cdots x_n$ y e coincide cada cadena x_i.

3.1.2. Algoritmos

Vamos a explicar los distintos algoritmos sobre los que Nrgrep está construido para permitir las búsquedas en los distintos tipos de patrones. Comenzaremos describiendo los algoritmos como Shift-And y BDM [35, 34]. Para luego centrarnos en BNDM que es el adoptado por Nrgrep

Paralelismo de bits El paralelismo de bits [36] utiliza una técnica aprovechando el paralelismo intrínseco de las operaciones de bits dentro de la palabra (word) de computadora. El número de operaciones que un algoritmo realiza se puede reducir por un factor como máximo w, donde w es el número de bits de la palabra de la computadora. La figura 3.3 muestra un autómata no determinista que busca un patrón dentro de un texto. El algoritmo Shift-Or [14] utiliza el paralelismo de bits para simular el autómata en su forma no determinista. Toma tiempo $O(\frac{mn}{w})$ en el peor caso.

Algoritmo Shift-And El algoritmo Shift-And [32] es una variante del algoritmo Shift-Or. Dado un patrón P = $p_1 p_2 \cdots p_m$ $p_i \in \Sigma$ y un texto T = $t_1 t_2 \cdots t_n$ $t_i \in \Sigma$, el algoritmo construye una tabla B donde para cada caracter guarda una máscara de bits $b_m b_{m-1} \cdots b_1$. La máscara en $B[c]$ tiene el iésimo bit seteado si

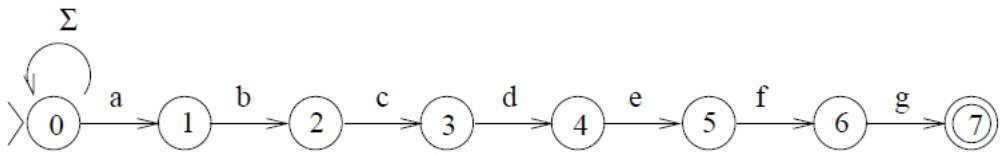

Figura 3.3: Autómata no determinista (afnd) para búscar el patrón P = "abcdefg" en un texto

sólo si p_i = c. El estado de la búsqueda se almacena en una palabra de máquina D = $d_m d_{m-1} \cdots d_1$ donde d_i es seteado siempre y cuando $p_1 p_2 .. p_i$ coincide con el final del texto leído hasta el momento. Por lo tanto se roporta una coincidencia siempre que d_m está seteado.
Se setea D = 0^m originalmente y para cada caracter nuevo del texto t_j se actualiza D con la siguiente fórmula.

$$D' \leftarrow ((D << 1) \mid 0^{m-1}\ 1)\ \&\ B[t_j]$$

El costo de este algoritmo es *0(n)*. Para patrones más largos que la palabra de computadora, en el peor caso es $O(\frac{mn}{w})$ y en promedio es *O(n)*.

El algoritmo BDM La gran desventaja del algoritmo Shift-Or es que no permite omitir caracteres. Alternativamente BDM si lo permite.
El algoritmo BDM está basado en un autómata sufijo (automaton suffix). Un autómata sufijo en un patrón P = $p_1 p_2 \cdots p_m$ es un autómata que reconoce todos los sufijos de P. Una versión no determinista de este autómata (AFND) tiene una estructura muy regular como se muestra en la figura 3.4. En el algoritmo BDM este autómata se hace determinista.
Un hecho importante es que este autómata se puede utilizar no sólo sufijos de P sino que también factores de P. Notar que hay un camino etiquetado como x desde el estado inicial, si sólo si, x es un factor de P. Esto quiere decir, que el AFND no se quedará sin estados activos hasta que no haya leído un factor de P. El autómata sufijo es usado para diseñar un algoritmo para búsqueda de patrones simples. Este algoritmo corre en tiempo *O(mn)* en el peor de los casos, pero es óptimo en caso promedio con $O(\frac{n log_\sigma m}{m})$. Para buscar un patrón $P = p_1 p_2 \cdots p_m$ en un texto T = $t_1 t_2 \cdots t_n$ el autómata sufijo de $P^r = p_m p_{m-1} \cdots p1$ se construye. Una ventana de largo m se desliza a lo largo del texto de izquierda a derecha. El algoritmo busca hacia atrás dentro de la ventana para un factor del patrón P usando el autómata sufijo. Esta búsqueda hacia atrás tiene dos posibles finales.

1. Falla en reconocer un factor, es decir, se llega a un caracter de ventana σ que hace que el autómata se quede sin estados que activar. Esto significa que el sufijo de la ventana que leímos no es más un factor de P. La figura 3.5 ilustra este caso. Luego movemos la ventana hacia la derecha con posición inicial correspondiente a la posición siguiente al caracter σ.

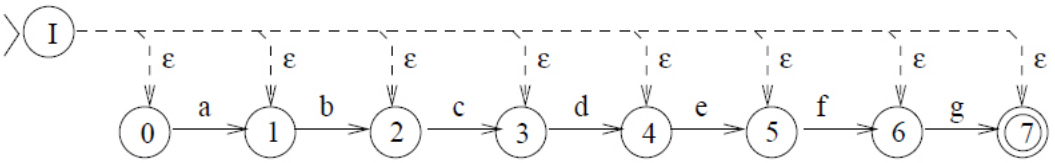

Figura 3.4: Autómata sufijo no determinista para búscar el patrón P = "abcdefg". Líneas discontinuas representa $\epsilon-$transiciones

2. Se llega al principio de la ventana, reconociendo así el patron P, dado que la ventana de largo m es un factor de P. Se reporta la ocurrencia y movemos la ventana una posición.

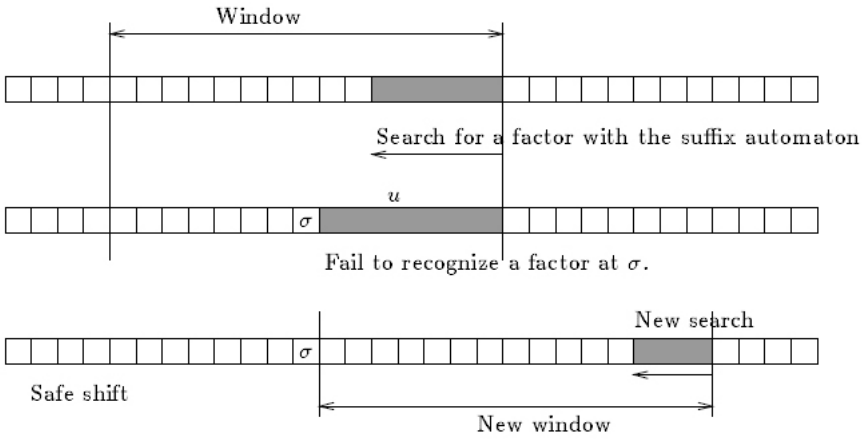

Figura 3.5: Búsqueda de autómata sufijo

Algoritmo BNDM: combinando el algoritmo Shift-Or y el BDM Describimos el algoritmo BNDM [22]. Este algoritmo (combinación del algoritmo Shift-Or y BDM) tiene todas las ventajas del escaneo hacia atrás con paralelismo de bits y además es capaz de omitir algunos caracteres como el algoritmo BDM.
En vez de hacer el autómata de la figura 3.4 determinista, BNDM lo simula usando paralelismo de bits. El paralelismo de bits funciona de la siguiente manera. Al igual que para Shift-And se guarda el estado de la búsqueda usando m bits de la palabra de computadora $D = d_m \cdots d_1$. Cada vez que se posiciona la ventana en el texto se inicializa D = 1^m y se escane la ventana hacia atrás.
Para cada caracter nuevo leído en la ventana se actualiza D. Si ya no hay más 1's en D entonces no puede haber una coincidencia y suspendemos el escaneo y movemos la ventana. En caso contrario, si podemos hacer m iteraciones entonces se reporta una coincidencia.

Se usa una tabla B donde para cada caracter c se guarda una máscara de bits. La fórmula para actualizar D es la siguiente.

$$D' \leftarrow (D \ \& \ \ B[t_j]) \ << \ 1$$

BNDM no sólo es más rápido que Shift-Or y BDM sino que además puede hacer búsqueda para los distintos tipos de patrones mencionados anteriormente. En particular, puede fácilmente lidiar con clases de caracteres solamente alterando el procesamiento y por lo tanto es el mejor algoritmo para este tipo de patrones [22, 23].

Búsqueda aproximada Una búsqueda aproximada significa encontrar las subcadenas que pueden ser convertidas en el patrón realizando como máximo k "operaciones" en ellas. Permitir un número limitado k de dichas diferencias (errores) es fundamental para la búsqueda que realizamos en este trabajo. Vamos a tener en cuenta tres tipos de errores: inserciones de caracteres, deleciones de caracteres y reemplazo de caracteres por otro (sustitución). Se puede utilizar tres diferentes ideas sobre esto que las detallamos a continuación.

Búsqueda hacia adelante La idea más básica que se adapta bien al paralelismo de bits es tener *k+1* autómatas similares representando el estado de la búsqueda con *cero* hasta k errores ocurrieron. Además de las flechas propias de los autómatas hay flechas que van del autómata i al *i+1* correspondiente a los distintos errores.
La figura 3.6 muestra un ejemplo con $k = 2$ errores. Cada fila representa el número de errores obtenidos. La primera fila representa cero errores, la segunda 1 error y así sucesivamente. Cada una de las columnas representa la coincidencia de un prefijo del patrón. Cada vez que se lee un carácter del texto el autómata cambia de estado. Las flechas horizontales, verticales, diagonales sólidas y diagonales punteadas, representan la coincidencia, inserción, sustitución o eliminación de un carácter respectivamente. El ciclo en el estado inicial permite que una coincidencia ocurra en cualquier parte del texto. El autómata indica una coincidencia cuando algún estado final está activo, la fila en la que se encuentre ese estado indicará el número de errores encontrados.
Se extiende una simulación simple para búsqueda aproximada con paralelismo de bits. Generalmente los algoritmos de paralelismo de bits simulan los algoritmos clásicos, algunos paralelizan el cálculo de la matriz de programación dinámica y algunos paralelizan el cálculo del AFN. La técnica más simple enfocada a paralelizar el cálculo del AFN, empaqueta cada fila i del AFN en diferentes palabras de computadora R_i donde cada estado está representado por un bit. Cada vez que se lee un carácter del texto, todas las transiciones del autómata se simulan usando operaciones de bits entre las *k +1* máscaras de bits, las cuales tienen la misma estructura, es decir, el mismo bit está alineado a la misma posición del texto. Para actualizar los valores de R'_i en la posición del texto j teniendo los

valores actuales R_i se aplica la siguiente fórmula:

$$R'_0 \leftarrow ((R_0 << 1) \mid 0^{m-1}1) \,\&\, B[t_j] \tag{3.1}$$

$$R'_1 \leftarrow ((R_i << 1) \,\&\, B[t_j] \mid R_{i-1} \mid (R_{i-1} << 1) \mid (R'_{i-1} << 1) \tag{3.2}$$

Donde B es una tabla que almacena una máscara de bits $b_m \cdots b_1$ para cada

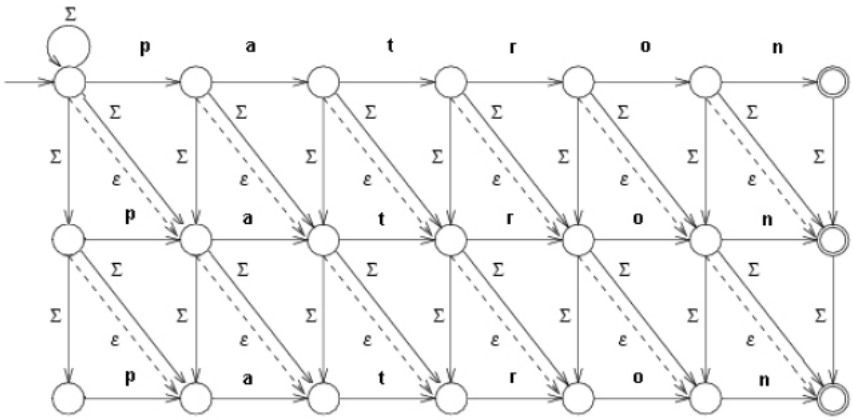

Figura 3.6: Autómata finito no determinístico para búsqueda aproximada de la cadena "patrón" permitiendo dos errores

carácter del patrón. La máscara en *B[c]* tiene el j^{th} bit activo si $p_j = c$. La búsqueda se inicia con $R_i = 0^{m-i}\ 1^i$. En la fórmula, R'_i expresa las flechas horizontales, verticales, diagonales sólidas y diagonales 3.6 respectivamente.

Búsqueda hacia atrás La búsqueda hacia atrás se puede adaptar fácilmente del autómata de búsuqda hacia adelante siguiendo la misma técnica usada para búsqueda exacta [22, 23]. Esto es, se construye un autómata como el de la figura 3.6 para el patron inverso, considerando todos los estados como 1 iniciales y considerando como único estado final al primer nodo de la última fila. Esto reconoce todos los prefijos inversos de P permitiendo como máximo k errores y tendrá estados activados siempre y cuando algún factor de P se haya visitado.

Dividiendo en k+1 subpatrones Una propiedad conocida [16, 32] establece que bajo un modelo con inserciones, deleciones y substituciones si el patrón se divide en *k+1* piezas continuas, entonces al menos una de las piezas se encontrará sin errores dentro de cualquier coincidencia con máximo k errores. Esto es fácil de verificar ya que cada operación puede alterar como máximo una pieza. Entonces la técnica consiste en realizar ua búsqueda en patrones múltiplos para las piezas sin errores y chequeando el texto que rodea las ocurrencias de cada pieza para una completa ocurrencia aproximada de todo el patrón. Ahora estamos

frente a un problema de búsquedas de patrones múltiples. Esto se puede resolver mediante una simple modificación del algoritmo de búsqueda hacia atrás para patrones simples [22, 23].

3.1.3. Búsqueda para patrones simples

Nrgrep utiliza el algoritmo BNDM cuando hace la búsqueda para patrones simples. Aunque se le agregan las siguientes modificaciones, necesarias para convertir el algoritmo de *pattern matching* en un software de búsqueda de patrones.

- Salida orientada al registro: la forma más útil de presentar los resultados es imprimiendo un contexto de la porción de texto que coincide con el patrón. El texto se considera una secuencia de registros y cuando se encuentra una coincidencia se imprime todo el registro. Un limitador de registro puede ser un fin de línea.
- Buffer de texto: se necesita para enfrentar el problema de archivos grandes y para lograr un óptimo rendimiento.
- Contexto: otra carácteristica del nrgrep es la especificación del contexto. Esto significa que la ocurrencia del patrón tiene que estar rodeada de ciertos caracteres para que sea considerado como tal.
- Seleccionar el patrón óptimo de exploración: se diseñó un algoritmo general que, bajo la presunción que los caracteres del texto son independientes, encuentra el mejor subpatrón en el peor de los casos en tiempo $O(m^3)$, aunque en la práctica es cercano a $O(m^2 log m)$.

3.1.4. Búsqueda para patrones extendidos

En este caso se permite caracteres opcionales y repetibles. Cada uno de estos caracteres se trata de manera distinta y todos se integran en un autómata más general que el de la figura 3.3. Sobre este autómata luego se aplica la maquinaria general de búsqueda hacia atrás. Los distintos caracteres que se permiten son:

- Especificar clases (o caracteres) opcionales: la notación que se utiliza es agregar un símbolo "?" luego del carácter o clase que se quiere hacer opcional.
- Permitir repeticiones de clases (o caracteres): la notación que se utiliza es agregar un símbolo "*" luego del carácter o clase y significa que la clase puede aparecer cero o más veces y el símbolo "+" luego del carácter o clase y significa que la clase puede aparecer una o más veces.

Se extiende el enfoque de búsqueda hacia adelante y hacia atrás usada para patrones simples para incluir patrones con símbolos opcionales. Para la verificación de ocurrencias se hace una pequeña modificación de patrones al igual que la selección de un buen patrón de exploración.

3.1.5. Búsqueda para expresiones regulares

Los patrones más complejos con los que nrgrep puede realizar la búsqueda son las expresiones regulares. Para esto se utiliza una técnica que extiende el algoritmo Shift-Or de dos maneras distintas [32, 25, 26] usando un autómata finito no determinista a partir de la expresión regular. En la figura 3.7 se muestra dos construcciones distintas para el patrón "$abcd(d|\epsilon)(e|f)de''$ utilizando las construcciones de Thompson [21] y Berry [28] de AFND para expresiones regulares. No lo detallaremos en este trabajo debido a que por el momento no utilizamos expresiones regulares para hacer las búsquedas que nos interesan.

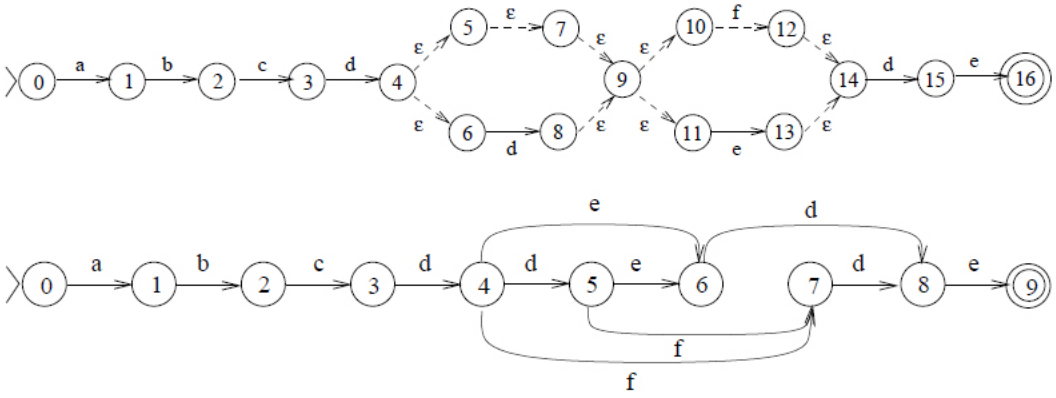

Figura 3.7: Resultado de AFND de Thompson (top) y Glushkov (botton) para la expresión regular "$abcd(d|\epsilon)(e|f)de''$

3.1.6. Pattern Matching aproximado

Además de los distintos tipos de patrones utilizados, tenemos la búsqueda aproximada. Mediante esta búsqueda consideramos tener un número limitado de errores. Permitiendo al menos k inserciones, deleciones o sustituciones. El usuario especifica el subconjunto de estos tres errores permitidos. De todos modos, diseñar un algoritmo diferente para cada subconjunto era poco práctico y en contra del espíritu de un software uniforme. Para los tres tipos distintos de patrones se añadieron diferntes características.

Patrones simples

Se hace uso de las tres técnicas descritas en la sección 3.1.2. Eligiendo la que promete ser mejor.

- Búsqueda hacia adelante: en esta búsqueda *k+1* filas corresponden al patrón que se usa. Existe un algoritmo de paralelismo de bits para simular el autómata en caso de k inserciones, deleciones y sustituciones [32]

- Búsqueda hacia atrás: se puede adaptar desde la técnica al igual que el algoritmo básico. Un punto sutil es que no se considera el autómata apagado hasta que ambas máscaras de R y T se queden sin estados activos, ya que T puede despertar a un R apagado. Como antes, se selecciona el mejor subpatrón para buscar que es como máximo de largo *w*. El algoritmo para elegir el mejor subpatrón tiene en cuenta que se permiten errores.

- Dividiendo en k+1 subpatrones: se puede adaptar al igual que trabajos anteriores, teniendo en cuenta dejar un hueco entre cada par de piezas.

Patrones extendidos

El tratamiento para patrones extendido es una combinación de la extensión de patrones simples a patrones extendidos sin errores y el uso de *k+1* filas o la partición en *k+1* piezas para permitir *k* errores. Aunque existen algunas complicaciones menores.

Expresiones regulares

Finalmente nrgrep permite hacer búsquedas con expresiones regulares permitiendo errores. El mismo mecanismo usado para patrones simples y patrones extendidos es utilizado. Es decir utilizando *k+1* réplicas del autómata de búsqueda y particionando al patrón en *k+1* piezas disjuntas. Ambas adaptaciones presentan complicaciones importantes con respecto a las adaptaciones simples.

- Búsqueda hacia atrás y hacia adelante: un problema en las expresiones regulares es que el concepto de "adelante" no significa inmediatamente un movimiento hacia la derecha. Búsqueda aproximada con *k+1* copias de AFND de las expresiones regulares implica poder moverse "hacia adelante" desde la fila *i* hacia la fila *i+1s*. La búsqueda hacia atrás y la búsqueda hacia adelante son llevadas a cabo en la forma normal usando una técnica distinta para realizar la actualización. La técnica para elegir el mejor factor necesario se mantiene inalterada, excepto que se agrega *k* al número de caracteres que se escanean dentro de cada ventana de texto.

- Dividiendo en k+1 subpatrones: debido a la complejidad y a que es extenso para desarrollarlo en este trabajo no vamos a detallar este problema, en caso de interés se puede ver más en [17].

3.1.7. Software Nrgrep

Finalmente se describe el software nrgrep. Fue desarrollado en ANSI C y testeado en plataforma Linux y Solaris. El código fuente esta disponible bajo licencia GNU[1].

[1]http://www.dcc.uchile.cl/ gnavarro/pubcode/

Uso y opciones

Nrgrep recibe por línea de comando un patrón y una lista de archivos para hacer la búsqueda. Si no se especifica ninguna opción, nrgrep hace la búsqueda en el orden dado e imprime todas las líneas donde encuentra los patrones. Las opciones son las siguientes:

- i: la búsqueda es insensible a mayúsculas y minúsculas;
- w: sólo palabras completas que coinciden con el patrón son aceptadas;
- x: sólo registros (ej:líneas) completos que coinciden con el patrón aceptados;
- c: sólo contar las coincidencias, no imprimirlos ;
- l: mostrar los nombres de los archivos que contienen coincidencias, pero no mostrar las coincidencias;
- G: mostrar todo el contexto de los archivos que contienen coincidencias;
- h: no mostrar nombres de archivos;
- n: imprimir registros precedido por sus números de registro;
- v: invertir la coincidencia, reportar los registros que no coinciden.
- $< delim >$: setear los limitadores de registros ($\backslash n$ por defecto) ;
- $< bufsize >$: setear el tamaño del búfer (64Kb por defecto);
- $< sep >$: imprimir la cadena $< sep >$ entre cada pares de registros mostrados;
- $< err >$ [ids]: permite hasta $< err >$ errores en las búsquedas. si la opción *ids* no está presente se permiten errores de: (i)nserciones, (d)eleciones y (s)ustituciones, caso contrario un subconjunto de ellos se puede especificar;
- L: tomar el patrón literalmente (sin caracteres especiales);
- H: ayuda.

Analizador semántico del patrón

Un aspecto importante es que se necesita analizar el patrón. En principio se analiza las expresiones regulares. Pero el módulo analizador lleva a cabo otras tareas como: analizar la sintaxis extendida, mapear el patrón a posiciones de máscara de bits, determinar tipo de subexpresiones y simplificaciones algebraicas.

3.2. PatMatch

PatMatch[30] fue adaptado a medida a partir de Nrgrep para ser usado como búsquedas de secuencias de ADN o de Proteínas. Es un herramienta de *pipeline* que chequea la sintaxis del patrón de entrada y luego utiliza el algoritmo Nrgrep [17], escrito en C también bajo Licencia Pública General GNU, para realizar la búsqueda. Una versión independiente del software se puede adaptar para usar con cualquier conjunto de datos que contenga secuencias y está disponible libre para su uso[2].

PatMatch fue diseñado para buscar secuencias cortas (entre 3-30 nt). Es útil para encontrar patrones cortos como por ejemplo ARNs pequeños. PatMatch usa una secuencia corta o una expresión regular como entrada y permite secuencias ambiguas representada con nomenclatura estándar IUPAC[3]. El programa también permite *mismatches* de la secuencia consultada para patrones en forma literal o con expresiones regulares.

Asimismo, soporta consultas para secuencias exactas y aproximadas empleando una sintaxis para expresiones regulares que incluye tanto como caracteres simples o ambiguos y expresiones regulares.

3.2.1. Patmatch como una adaptación de nrgrep

Nrgrep (que fue descrita anteriormente) es una herramienta general para búsqueda aproximada de patrones, es por esto que se agregó una *wrapper* en Perl denominado *scan_pipeline* que fue escrito para adaptar a Nrgrep en un programa enfocado en hacer búsquedas para nucleótidos y péptidos. El script chequea la sintaxis y traduce los patrones representados por la nomenclatura estándar IUPAC en el conjunto de nucleótidos o aminoácidos que representan. La sintaxis del patrón de entrada en PatMatch también fue convertida en una expresión regular diferente de la que utiliza Nrgrep. Por ejemplo el patrón que representada la búsqueda de tres a cinco ocurrencias de la subsecuencia MWA en una secuencia de péptidos es $WMA\{3,5\}$ en la sintaxis de PatMatch y $[(MWA)(MWA)(MWA)(MWA)?(MWA)?]$ en la sintaxis de Nrgrep.

3.2.2. Comparación con herramientas similares

Existen dos tipos de algoritmos de búsqueda de patrones comunmente usados en biología estos son $scan\_for\_matches$ y programas tipo $grep\_like$. PatScan [24, 15] es un programa basado en $scan\_for\_matches$ que ha añadido características tales como significación estadística del patrón mediante una simulación de cadenas de Markov.

Otro grupo de software para encontrar patrones de ADN y proteínas estan basados en *grep*. Una herramienta $grep\_like$ llamada tacg [6] soporta expresiones regulares, degeneraciones IUPAC, búsqueda con errores y matrices de probabilidades. Como Patmach no soporta búsqueda con matrices de probabilidades tacg no soporta inserciones ni deleciones. Adicionalmente, tacg tiene la desventaja de ser más lento que la familia

[2] ftp://ftp.arabidopsis.org/home/tair/Software/Patmatch/

[3] http://www.chem.qmw.ac.uk/iupac

de herramientas *grep*, así como un algoritmo ineficiente al buscar patrones degenerados [6]. *eMOTIF-SCAN* es un programa que utiliza la herramienta *agrep* que soporta patrones aproximados y expresiones regulares contra la bases de datos de motivos de proteínas de eMOTIF [18]. Patmtach tiene una ventaja sobre eMOTIF, ya que utiliza Nrgrep que fue mostrado como la herramienta más rapida para búsqueda de cadenas [17].

Los programas *fuzznuc* y *fuzzpro* disponibles en EMBOS[4](European Molecular Biology Open Source Software Suite) provee búsqueda de patrones para secuencias de proteínas y de nucleótidos. El inconveniente es que estos dos programas no soportan repeticiones para grupos de nucleótidos o aminoácidos. Esta era una característica deseada de PatMatch y fue una razón por la fue elegido Nrgrep sobre *fuzznuc* y *fuzzpro*

3.3. PatmatchMicro

En esta sección vamos a describir la herramienta final de la primer etapa denominada PatmatchMicro como mostramos anteriormente en la figura 3.2. Describiendo primero la entrada del programa, para luego mostrar los módulos desarrollados. Estos módulos se utilizan para automatizar una búsqueda de genes blancos de microARNs más compleja que incluya múltiples bases de datos de plantas donde se efectuará dicha búsqueda. Además se agregaron nuevos parámetros variables que se describen a continuación y se integraron, a la herramienta Patmatch, dichos módulos siguiendo una arquitectura en *pipeline*.

3.3.1. Entrada de PatmatchMicro

Una de las entradas del PatmatchMicro son las secuencias de los microARNs conservados tomados en este trabajo. De esta manera, seleccionamos tomando la más representativa una secuencia mínima consenso para cada familia de microARN conservado que se encuentran en miRBase 15.0[5]. En este sitio, existe una base de datos disponible para su uso libre que provee información de todas las secuencias de microARNs identificadas hasta el momento para un amplio número de genes.

Para cada especie de planta existen distintas secuencias que difieren en composición en algunos pocos nucleótidos (nt) entre ellas y su longitud varía entre 20 nt y 23 nt. Además en algunas especies existen distintas secuencias que identifican a un mismo microARN. Descartamos el primer nucleótido y los n restantes del final hasta obtener una secuencia consenso de 18 nucleótidos. En la figura 3.8 se muestra un ejemplo para un microARN en particular denominado miR164 en distintas especies de plantas donde está presente ese microARN.

Luego con estas secuencias son utilizadas para dar comienzo a la estrategia de búsquedas de genes blancos. Las utilizaremos como entrada de la herramienta Pat-

[4]http://emboss.sourceforge.net/

[5]http://mirbase.org/

Arabidopsis Thaliana	miR164a	UGGAGAAGCAGGGCACGUGCA
	miR164b	UGGAGAAGCAGGGCACGUGCA
	miR164c	UGGAGAAGCAGGGCACGUGCG
Oryza Sativa	miR164a	UGGAGAAGCAGGGCACGUGCA
	miR164b	UGGAGAAGCAGGGCACGUGCA
	miR164c	UGGAGAAGCAGGGUACGUGCA
	miR164d	UGGAGAAGCAGGGCACGUGCU
	miR164e	UGGAGAAGCAGGGCACGUGAG
	miR164f	UGGAGAAGCAGGGCACGUGCA
Populus trichocarpa	miR164a	UGGAGAAGCAGGGCACGUGCA
	miR164b	UGGAGAAGCAGGGCACGUGCA
	miR164c	UGGAGAAGCAGGGCACGUGCA
	miR164d	UGGAGAAGCAGGGCACGUGCA
	miR164e	UGGAGAAGCAGGGCACGUGCA
	miR164f	UGGAGAAGCAGGGCACAUGCU
Consenso		**GGAGAAGCAGGGCACGUG**

Figura 3.8: Secuencia consenso del miR164

matchMicro para predecir nuevos genes blancos, así como también como entrada para la herramienta final denominada Target3VU (ver figura 3.1).

Siendo el alfabeto del microARN

$$\Sigma = \{\text{A,C,G,U}\}.$$

Consideremos este alfabeto. Entonces la entrada de PatmatchMicro puede ser una secuencia entera de un microARN (de aproximado 20 nt de longitud). Y particularmente puede ser una secuencia de la forma:

$$miR = \alpha_1\alpha_2...\alpha_{18} \;\; con\; \alpha_i \in \Sigma \;,\; i = 1...18$$

Estas cadenas, miR son de longitud constante igual a 18, que es largo de las secuencias consenso que nosotros consideramos. Esta búsqueda se repetirá para los 22 microARNs conservados que se detallan a continuación en la figura 3.9 el nombre que lo identifica estará compuesto de miR más un número que se asigna de forma secuencial, del mismo modo que se los describe en la literatura.

3.3.2. Parámetros y módulos implementados

En lo que sigue, describimos los parámetros variables de entrada, así como los módulos que fueron integrados para permitir una búsqueda además de flexible más

microARN	Consenso
miR156	GACAGAAGAGAGUGAGCA
miR159	UUGGAUUGAAGGGAGCUC
miR160	GCCUGGCUCCCUGUAUGC
miR162	CGAUAAACCUCUGCAUCC
miR164	GGAGAAGCAGGGCACGUG
miR166	CGGACCAGGCUUCAUUCC
miR167	GAAGCUGCCAGCAUGAUC
miR168	CGCUUGGUGCAGGUCGGG
miR169	AGCCAAGGAUGACUUGCC
miR171	GAUUGAGCCGCGCCAAUA
miR172	GAAUCUUGAUGAUGCUGC
miR319	UGGACUGAAGGGAGCUCC
miR390	AGCUCAGGAGGGAUAGCG
miR393	CCAAAGGGAUCGCAUUGA
miR394	UGGCAUUCUGUCCACCUC
miR395	UGAAGUGUUUGGGGGAAC
miR396	UCCACAGCUUUCUUGAAC
miR397	CAUUGAGUGCAGCGUUGA
miR398	GUGUUCUCAGGUCACCCC
miR399	GCCAAAGGAGAUUUGCCC
miR408	UGCACUGCCUCUUCCCUG
miR827	UAGAUGACCAUCAGCAAA

Figura 3.9: Entrada

completa. Estos módulos son RNAHybrid, Alineamientos y Blast. Y fueron integrados en la herramienta PatmatchMicro (ver figura 3.1).

Mismatches

Sabemos que el apareamiento o complementariedad de bases entre microARN y su secuencia blanco se da de la siguiente forma. La adenina y uracilo son complementarias o la adenina con la timina (A=U) o (A=T) dependiendo si es ADN o ARN. Del mismo modo la guanina y citosina también lo son (G=C).

De esta manera decimos que una secuencia microARN y otra secuencia blanco tienen apareamiento perfecto, si todas las bases de la secuencia del microARN se complementan de manera perfecta uno a uno con las bases de la secuencia blanco. Como en la figura 3.10.

Cuando esto no ocurre decimos que la secuencia tiene desapareamientos o *mismatches*. Existen tres tipos diferentes de *mismatches* denominados **inserción**, **deleción** y

Figura 3.10: Apareamiento perfecto

sustitución. Como se muestra en la figura 3.11.

Figura 3.11: Mismatches

Sabemos que existen genes blancos que se validaron experimentalmente en otros trabajos donde la interacción del par microARN-gen blanco permite alguna cantidad y tipo limitado de *mismatches* [8]. Por esto, permitimos hasta 3 *mismatches* de tipo inserción y sustitución. Es decir que no permitimos deleciones en la secuencia blanco ya que esta no es una interacción microARN-gen blanco permitida.

Módulo blast

Las búsquedas de genes blancos las hicimos sobre bases de datos de ciertas especies de plantas que presentan una escasa descripción del tipo genómica funcional, es decir que no se sabe en que función biológica están involucrados dichos genes que componen las bases de datos. En cambio para la mayoría de los genes del genoma de *Arabidopsis thaliana* se tiene una descripción funcional ya sea experimental o computacional.

Integramos a la herramienta PatmatchMicro un módulo automatizado que usa el programa denominado BLAST (Basic Local Alignment Search Tool). Es un programa de alineamiento de secuencias de tipo local ya sea de ADN o proteínas. Integramos una variante llamada Blastx que utiliza como entrada una secuencia de nucleótidos. BLASTx traduce la secuencia en sus seis posibles marcos de lectura (tres marcos de lecturas por hebra) y compara estas secuencias traducidas contra una base de datos de proteínas. Esta herramienta se usa cuando se tiene sospecha de que la secuencia de entrada codifica para una proteína pero no se sabe exactamente cuál es su producto. El programa asigna para los mejores genes candidatos un determinado puntaje y un parámetro E conocido como "e-value" de corte que permite definir qué alineamientos queremos obtener de acuerdo a su significado estadístico. Cuanto menor sea el valor de E, más significativo es un alineamiento. Usamos un corte con un E menor a $1e-6$. Luego decimos que el gen buscado para una especie es homólogo al que aparece como primer candidato en *Arabidopsis thaliana*.

De esta manera tomamos cada gen de cada especie de plantas como en el ejemplo de la figura 3.13 e hicimos un blastx contra el genoma de *Arabidopsis thaliana* identificando de esta manera al gen mas similar que existe en Arabidopsis. Cada gen de Arabidopsis se enuentra identificado por un "locus ID" (identificador único para cada gen) y contiene un descripcion de su función. Por lo tanto, podemos usar el locus ID del gen más homólogo en Arabidopsis para proveer de una etiqueta (TAG) para identificar al candidato de otra especie, así como la descripción de la función del gen en *Arabidopsis thaliana*.

Entonces cada gen de las distintas especies y que no poseen una descripcion funcional, pueden tener asignada una etiqueta en base al gen más similar que se encuentre en *Arabidopsis*. Entonces en el ejemplo de la figura 3.13 lo que nos estaría diciendo es que el gen LOC_Os02g45570 de *Oryza Sativa* es homólogo al gen de *Arabidopsis thaliana* que tiene como "locus ID" AT3G52910.1. Podemos entonces usar a AT3G52910.1 para etiquetar al gen de arroz. Además tenemos una idea de su función biológica, y en el ejemplo el gen pertenece a una familia llamada GRF (growth-regulating factor).

>LOC_Os02g45570

```
ATGGATGAGGAGAAGGAAGCCGACTCGCCGCAGCCACCGTCCAAGCTGCCTCGCCTCTCCGGCGCTGACCCGAATGCCGGAGTGGTGACCATGGCAGCACC
CCCGCCGCCGGTGGGTCTTGGGCTGGGGCTTGGACTCGGCGGCGACAGCCGCGGCGAGCGTGACGTGGAAGCGTCGGCGGCGGCGGCGCACAAGGCGACGG
CGCTGACGTTCATGCAGCAGCAGGAGCTGGAGCACCAGGTGCTCATCTACCGCTACTTCGCCGCGGGCGCGCCCGTGCCGGTGCACCTCGTGCTCCCCATC
TGGAAGAGCGTCGCGTCCTCCTCCTTCGGCCCGCACCGCTTCCCTTCCCTGGCAGTGATGGGGTTGGGGAACCTGTGCTTCGACTACCGGAGCAGCATGGA
GCCGGACCCAGGGCGGTGCAGGCGCACGGACGGCAAGAAGTGGCGGTGCTCGCGCGACGTGGTGCCGGGGCACAAGTACTGCGAGCGGCACGTCCACCGCG
GACGCGGCCGTTCAAGAAAGCCTGTGGAAGCCTCCGCGGCCGCCACCCCGGCGAACAACGGCGGCGGCGGTGGCATCGTCTTCTCCCCCACCAGCGTCCTC
CTCGCCCACGGCACCGCGCGCGCCACCTGA
```

Figura 3.12: Secuencia del gen LOC_Os02g45570 de *Oryza Sativa*

Sequences producing significant alignments:	Score (bits)	E Value
AT3G52910.1 | Symbols: AtGRF4, GRF4 | growth-regulating factor	125	2e-29
AT4G37740.1 | Symbols: AtGRF2, GRF2 | growth-regulating factor	115	3e-26
AT2G35230.1 | Symbols: IKU1 | VQ motif-containing prot.	28	3.8

Figura 3.13: Mejores candidatos obtenidos utilizando Blastx

Como esto demanda mucho tiempo computacional hicimos un precálculo computando cada gen individual de las bases de datos a usar y asignándole un único mejor candidato que guardamos en una base de datos (denominada Blast) para su posterior acceso mediante la herramienta Target3VU como se mostró en la figura 3.1. Cuando el resultado era nulo para un gen le asignamos un valor NULL en la base de datos. Ya que para nuestro estudio no existe una homología fuerte con ningún gen en *Arabidopsis thaliana* por lo tanto para el análisis final de conservación evolutiva del sitio blanco éste gen no nos interesa.

Módulo RNAhybrid

Por otro lado, determinaremos la energía de interacción entre el microARN y el blanco implementando un nuevo módulo integrado, utilizando la herramienta "RNAhybrid"[6] que calculará de forma automatizada la menor energía libre de hibridación (mfe) entre el par microARN-gen blanco. Como se muestra en la figura 3.14. Cabe mencionar que las energías de interacción son negativas y mientras más grande es la energía en valor absoluto, más fuerte es la interacción. [29]

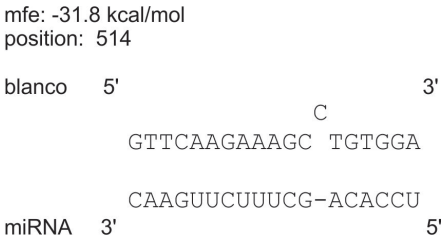

Figura 3.14: Mínima enrgía libre de hibridación microARN-gen blanco

Si m y n son el largo de la secuencia blanco y el microARN respectivamente y c es el largo máximo de un *loop* en cualquiera de las secuencias la complejidad espacial del algoritmo es del orden de $\Theta(\mathrm{mn})$ y la complejidad temporal es del orden de $\Theta(\mathrm{c}^2\mathrm{mn})$. Si m es mucho más largo que n y c, como es en el caso que estamos estudiando, la complejidad espacial y temporal es lineal en el largo de la secuencia blanco m.

Esta herramienta esta implementada a partir de la recurrencia de programación dinámica que se detalla a continuación.

$$
\begin{aligned}
H_{i,j} &= min\{0, T_{i,j}\,, C_{i,j}\,\} \\
T_{i,j} &= min\{T_{i+1,j}\,, B_{i,j}\,\} \\
B_{i,j} &= min\{B_{i,j+1}\,, \\
&\quad eds(i+1, j+1, C_{i+1,j+1}\,) \\
&\quad edt(i+1, j, C_{i+1,j}\,) \\
&\quad edb(i, j+1, C_{i,j+1}\,)\} \\
C_{i,j} &= if\ can\_pair(x_{i+1}, y_{j+1})\ then \\
&\quad min\{sr(i+1, j+1, C_{i+1,j+1}\,), \\
&\quad \min_{i+2\le k\le min(i+16,m-1)} \{bt(i+1, j+1, k, C_{k,j+1}\,)\}, \\
&\quad \min_{j+2\le l\le min(j+16,n-1)} \{bb(i+1, j+1, l, C_{i+1,l}\,)\}, \\
&\quad \min_{\substack{i+2\le k\le min(i+16,m-1)\\ j+2\le l\le min(j+16,n-1)}} \{il(i+1, j+1, k, l, C_{k,l}\,)\}, \\
&\quad el(i+1, j+1, m, n)\} \\
&\quad else\ \infty
\end{aligned}
$$

[6] http://bibiserv.techfak.uni-bielefeld.de/rnahybrid/

Con la secuencia blanco $x = x_1 \cdots x_m$ y el microARN $y = y_1 \cdots y_n$. La menor energía libre de hibridación (mfe) está en $H_{0,0}$, T(top) se usa para omitir las primeras bases de la secuencia blanco, B(bottom) se usa para omitir las primeras bases del microARN y C para subestructuras cerradas. Es decir $C_{i,j}$ es la mfe de las secuencias empezando en *i+1* y *j+1* respectivamente donde las bases x_{i+1} e y_{j+1} forman un par. *eds*, *edt* y *edb* son funciones de energía para simetría de bases colgando, una base superior colgando y una base inferior colgando respectivamente. *sr* es la energía de un par apilado *bt* y *bb* son las energías para el bulge, *il* es la energía de un *loop* interno y *el* la energía de un final abierto. Valores no definidos son ∞. Sólo C se necesita para lograr la mejor complejidad temporal. T y B son adicionalmente utilizados para acelerar el prodecimiento de rastreo (backtracking) que da la propia hibridación.

Debido a que las secuencias de microARNs son cortas, buenas MFEs pueden ocurrir por cuestiones de azar. Mientras más largas son las secuencias blancos mejores van a ser estas energías al azar. Entonces se normaliza la MFEs para eliminar la influencia del largo de las secuencias de la siguiente manera. Si e es la menor energía libre de hibridación, m es el largo de la secuencia blanco buscada y n el largo del microARNs, entonces la energía negativa normalizada e_n se define como sigue.

$$e_n = -\frac{e}{log(mn)}$$

Esta energía de hibridación la utilizaremos como un filtro, por medio de un corte, para descartar falsos positivos eliminando los pares microARN-gen blanco que no tienen una buena energía de hibridación según sugieren trabajos anteriores [31].

Móudulo alineamientos

El algoritmo de Needleman-Wunsch se utiliza para realizar alineamientos globales de dos secuencias. Los alineamientos globales, que intentan alinear cada nucleótido de cada secuencia, son más útiles cuando las secuencias iniciales son similares y aproximadamente del mismo tamaño. Nosotros implementamos en perl una modificación de este algoritmo de programación dinámica que siempre termina y garantiza que la solución devuelta sea óptima.

Unas de las razones de la modificación del algoritmo es que nosotros no permitimos deleciones en la secuencia blanco y además de mostrar las secuencias microARN-gen blanco mostramos el alineamiento obtenido de manera determinista.

Dadas dos secuencias A y B,

$$A = a_1, a_2, \cdots, a_m \; y \; B = b_1, b_2, \cdots, b_n$$

podemos expresar el problema con esta recurrencia

$$M_{i,j} = \begin{cases} M_{i-1,j-1} + S(a_i, b_j) \\ M_{i,j-1} - 1 \\ M_{i-1,j} - 1 \end{cases} \tag{3.3}$$

con

$$S_(a_i, b_j) = \begin{cases} 1 & si\ a_i = b_j \\ -1 & si\ a_i \neq b_j \end{cases}$$

Como el módulo está integrado a la herramienta PatmatchMicro el microARN y el blanco los tengo almacenado en dos variables, donde $mirna es el microARN y $target es la secuencia blanco, busco el reverso y complementario del microARN (ya que de esta forma es como hago el alineamiento).

```
$mirna = complement ($mirna);
$mirna = reverse $mirna;
```

Acá esta la funcion Needleman que hace el alineamiento.

```
Needleman($mirna,$target);
sub Needleman {
    (my $mir, my $targ) = @_;
    my $MATCH    =  1;
    my $MISMATCH = -1;
    my $GAP      = -1;
    my $INF = Math::BigInt->binf;
```

Para inicializar la matriz, completamos la primera fila y la primera columna. El *score* de cada entrada lo hacemos multiplicando la distancia del origen por -1 *(gap)* .

```
    my @matrix;
    $matrix[0][0]{score}   = 0;
    $matrix[0][0]{pointer} = "none";

    for(my $j = 1; $j <= length($miR); $j++) {
        $matrix[0][$j]{score}   = $GAP * $j ;
        $matrix[0][$j]{pointer} = "left";
    }
    for (my $i = 1; $i <= length($targ); $i++) {
        $matrix[$i][0]{score}   = $GAP * $i ;
        $matrix[$i][0]{pointer} = "up";
    }
```

La llenamos siguiendo la ecuación 3.3

```
for(my $i = 1; $i <= length($targ); $i++) {
    for(my $j = 1; $j <= length($miR); $j++) {
        my ($diagonal_score, $left_score, $up_score);
        my $letter1 = substr($miR, $j-1, 1);
```

```
            my $letter2 = substr($targ, $i-1, 1);

            if ($letter1 eq $letter2) {
                $diagonal_score = $matrix[$i-1][$j-1]{score} + $MATCH;
            }
            else {
                $diagonal_score = $matrix[$i-1][$j-1]{score} + $MISMATCH;
            }

            $up_score   = $matrix[$i-1][$j]{score} + $GAP;
            #no permitimos deleciones
            $left_score = $matrix[$i][$j-1]{score} + ($GAP*$INF);

            if ($diagonal_score >= $up_score) {
                if ($diagonal_score >= $left_score) {
                    $matrix[$i][$j]{score}   = $diagonal_score;
                    $matrix[$i][$j]{pointer} = "diagonal";
                }
                else {
                    $matrix[$i][$j]{score}   = $left_score;
                    $matrix[$i][$j]{pointer} = "left";
                }
            }
            else {
                if ($up_score >= $left_score) {
                    $matrix[$i][$j]{score}   = $up_score;
                    $matrix[$i][$j]{pointer} = "up";
                }
                else {
                    $matrix[$i][$j]{score}   = $left_score;
                    $matrix[$i][$j]{pointer} = "left";
                }
            }
        }
    }
```

Para recuperar el alineamiento desde la matriz empezamos en la esquina inferior derecha y terminamos en la esquina superior izquierda siguiendo el apuntador decidiendo si nos movemos hacia arriba o diagonal y chequeando si hay una coincidencia, una sustitución o una inserción. En nuestro caso al no permitir deleciones nunca voy a ir hacia la izquierda.

```
    my $miARN_res = "";
    my $target_res = "";
```

```
    my $align_res;

    my $j = length($miR);
    my $i = length($targ);

    while (1) {
        last if $matrix[$i][$j]{pointer} eq "none";
        my $letter1 = substr($miR, $j-1, 1);
        my $letter2 = substr($targ, $i-1, 1);

        if ($matrix[$i][$j]{pointer} eq "diagonal") {
            if ($letter1 eq $letter2){
                $align_res .= "|";
            }
            else
            {
                $align_res .= "*";
            }
            $miARN_res .= substr($miR, $j-1, 1);
            $target_res .= substr($targ, $i-1, 1);
            $i--;
            $j--;
        }
        elsif ($matrix[$i][$j]{pointer} eq "left") {
            $miARN_res .= substr($miR, $j-1, 1);
            $target_res .= "-";
            $align_res .= "*";
            $j--;
        }
        elsif ($matrix[$i][$j]{pointer} eq "up") {
            $miARN_res .= "-";
            $target_res .= substr($targ, $i-1, 1);
            $align_res .= "*";
            $i--;
        }
    }
```

Por ultimo tenemos el blanco, el alineamiento y el microARN en las variables $target_res $align_res y $miARN_res.

```
    $target_res = reverse $target_res;
    $align_res = reverse $align_res;
    $miARN_res = complement (reverse $miARN_res);
}
```

3.3.3. Especies

En general las secuencias de nucleótidos sufren mutaciones a lo largo del tiempo a no ser que haya una presión de selección en contra de que esto ocurra. Entonces si la regulación de un gen por un microARN es importante el sitio reconocido por el microARN debería estar conservado en distintas especies.

Es por esto que la estrategia para buscar nuevos genes blancos está basada en la conservación evolutiva del sitio blanco como muestra la figura 3.15. Por esta razón partimos de microARNs conservados en distintas especies de plantas que divergieron a lo largo de millones de años. Debido a esto la búsqueda la realizamos sobre un rango de especies donde estos microARNs están conservados [19]. Los genomas que utilizamos en este trabajo tienen tamaños muy variables, al igual que los genes que estan contenidos en dichos genomas. Estos pueden variar desde algunos KB hasta 200 MB dependiendo de cuan completas estén dichas bases de datos. Estos genomas son parseados de archivos en formato fasta como mostramos en la figura 3.1.

Especie	Sitio blanco
Arabidopsis Thaliana	GTTCAAGAAAGACTGTGGA
Brassica Napus	GTTCAGGAAAGCCTGTGGA
Glycine_max	GTTCAAGAAAGCTTGTGGA
Ipomoea_nil	GTTCAAGAAAGCATGTGGA
Oryza sativa	GTTCAAGAAAGCCTGTGGA

Figura 3.15: Conservación del sitio blanco para un gen denominado GRF

Las base de datos fueron obtenidos de "Gene Index Project"[7], un proyecto mantenido y administrado por la universidad de Harvard que contiene un catálogo completo de genes en una amplia gama de organismos incluyendo plantas. Uno de los problemas sobre estas bases de datos es que están incompletas. Pero tiene la ventaja de que estos genomas no son redundantes, es decir no hay información repetida. Además de estas bases de datos, se utilizarán ARNm completos para las especies cuyo genoma fue secuenciado y caracterizado. Como es el caso de *Arabidopsis thaliana* que su genoma fue secuenciado de manera completa en el año 2000.

El formato utilizado para representar secuencias de ácidos nucleicos se lo conoce en bioinformática como formato FASTA[8]. Los genes (y por lo tanto los genomas) que utilizamos en este trabajo se encuentran en dicho formato. La simplicidad del formato FASTA hace fácil el manipular y analizar secuencias usando herramientas de procesado de textos y lenguajes de script como Python y Perl. Es por eso que gran parte de este trabajo utilizamos Perl como lenguaje.

Una secuencia bajo formato FASTA comienza con una descripción en una única línea (línea de cabecera), seguida por líneas de datos de secuencia. La línea de descripción se distingue de los datos de secuencia por un símbolo ">" en la primera posición.

[7] http://compbio.dfci.harvard.edu/tgi/

[8] http://www.ncbi.nlm.nih.gov/blast/fasta.shtml/

La palabra siguiente a este símbolo es el identificador de la secuencia (locusID), y el resto de la línea es la descripción (ambos son opcionales). En este trabajo, en caso de no tener identificador de la secuencia nosotros le incluimos uno para su posterior identificación. No debería existir espacio entre el ">" y la primera letra del identificador. Se recomienda que todas las líneas de texto sean menores de 80 caracteres. La secuencia termina si aparece otra línea comenzando con el símbolo ">" que indica el comienzo de otra secuencia. Un ejemplo simple de una secuencia en el formato FASTA puede ser el que se muestra en la figura 3.16.

```
>AT1G12260.1 Symbols: VND4, EMB2749, ANAC007, NAC007
GATTGATAGGGAAAGAGAGAGAGATGAAAGAAAGTAAAATATATAATAGA
TTATTAGGACACGATTGTCATCTTTTGATTTGTGTCTTGTGTGCTCTCTC
TTTCTTCTCTTCCTCGATTGATCTTCTTTATATAACCCTACTCTCTTTCT
CTTTTCCCATTCTTTCTATCATTCTCCCTTTCTCTCTCGGGATCTGATCT
CTCTTTCCAGTAACCTATTCCCGAGGAGCACTGTCAAATCTTGTCCACTC
TTTGATCTTATCTCGATCTCTTTCTCTTTCTAGTCTTGTGTAGTCTTCAA
ACTTGTGATGTTATCTATATAGTAATCACGAGAGAGAATCATACAATAGC
TGAAACATAAAGCTTTCTTAGAAGCTTTAAAAAGGTCTCATCTGGATTAT
CCTGTTTAATTTCTAGAGTTTCTTCAGGCAGATTATTAACCGATCAAGAA
GACAAACATGAATTCATTTTCCCACGTCCCTCCGGGTTTTAGATTTCACC
CGACAGATGAAGAACTTGTAGACTACTACCTGAGGAAAAAAGTCGCATCG
AAGAGAATAGAAATTGATTTCATAAAGGACATTGATCTTTACAAGATTGA
GCCATGGGACCTTCAAGAGTTGTGCAAAATTGGGCATGAAGAGCAGAGTG
ATTGGTACTTCTTTAGCCATAAAGACAAGAAGTATCCCACAGGGACTCGA
ACCAATAGAGCAACAAAAGCAGGGTTTTGGAAAGCCACCGGAAGAGATAA
GGCTATCTATTTGAGGCATAGTCTAATTGGCATGAGGAAAACACTTGTGT
TTTACAAGGGAAGAGCCCCAAATGGACAAAAGTCTGATTGGATCATGCAC
GAATACCGCTTAGAAACCGATGAAAACGGAACTCCTCAGGAAGAAGGATG
GGTTGTGTGTAGGGTTTTCAAGAAGAGATTGGCTGCAGTTAGACGAATGG
GAGATTACGACTCATCCCCTTCACATTGGTACGATGATCAACTTTCTTTT
ATGGCCTCCGAGCTCGAGACAAACGGTCAACGACGGATTCTCCCCAATCA
TCATCAGCAGCAGCAGCACGAGCACCAACAACATATGCCATATGGCCTCA
ATGCATCTGCTTACGCTC
```

Figura 3.16: Gen denominado AT1G12260.1, perteneciente al genoma de emphArabidopsis thaliana

3.3.4. Salida de PatmatchMicro

Como salida de la herramienta PatmatchMicro obtuvimos distintos valores que luegon fueron almacenados en una base de datos (Genes Blancos) para su posterior analisis que detallamos en los capítulos posteriores como mostramos anteriormente en la figura 3.1. Un ejemplo acotado para describir los datos que se muestran por pantalla, es el que se muestra en la figura 3.17.

Donde

- *Specie*: es la especie de planta donde aparece el gen blanco que se obtiene como resultado de la búsqueda de un microARN, en este caso el mir398.
- *Sequence_name*: es el nombre con que se identifica al gen dentro del genoma de la planta donde fue hecha la búsqueda. Nosotros lo llamamos identificador o *LocusID*.

Specie	Sequence name	Matching Positions		5'-target-3' Alignment 3'-miRNA-5'	mfe	Best hit (score) (e-Value)	Description
		Start	End				
Oryza Sativa	LOC_Os12g07160.1	823	802	TGAAGGAATGTGTCAGGCAGCT ||||*|||||*|||*||||||| ACTTACTTACTCAG-CCGTCGA	ΔG: -28.08	AT1G77580.2 (278) (3e-15)	myosin heavy chain-related; similar to unknown protein [Arabidopsis thaliana] (TAIR:AT1G21810.1); similar to Protein of unknown function DUF869, plant [Medicago truncatula] (GB:ABE86171.1); similar to Os11g0170200 [Oryza sativa (japonica cultivar-group)] Protein of unknown function DUF869, plant;
Brassica Napus	Ev158747	523	450	TGACTGAATGTGTCAGGCAGCT |||*|||||*|||*||||||| ACTTACTTACTCAG-CCGTCGA	ΔG: -29.03	At4g27330 (478) (6e-21)	similar to UniRef100_Q2HZF8 Cluster: Sporocyteless; n=1; Brassica juncea|Rep: Sporocyteless - Brassica juncea (Leaf mustard) (Indian mustard), partial (30%)

Figura 3.17: Salida de PatmatchMicro para el miR398

- *Matching positions*: es la posición relativa dentro del gen donde aparece la secuencia blanco. Se muestra el comienzo y el fin de dicha secuencia dentro del gen identificado por su *LocusID*.
- *Alignment*: es el par microARN-gen blanco con su correspondiente alineamiento. Se muestra el alineamiento y en este caso la secuencia del microARN esta de esta forma $miR = \alpha_{18}\alpha_{17}...\alpha_1$, en forma inversa. Cuando esto ocurre decimos que la secuencia tiene dirección tres-cinco prima (del 3' al 5').
- *mfe*: es la mínima energía de hibridación calculada con el módulo RNAHybrid del par microARN-gen blanco.
- *Best hit*: Es el mejor candidato de *Arabidopsis thaliana* que aparece como resultado del módulo Blast, con su respectivo puntaje y E-value. Lo llamamos por su *LocusID* y sirve como etiqueta o TAG del gen en cuestión.
- *Description*: La descripción funcional en *Arabidopsis thaliana* del gen que da como resultado el módulo Blast que da como mejor candidato.

3.3.5. Base de datos

La salida del programa PatmatchMicro se puede ver en forma de tabla, gracias a una aplicación web que implementamos en Perl, como se muestra en la figura 3.17. De esta forma el usuario puede realizar la búsqueda, accediendo a un servidor web a través de una intranet mediante cualquier navegador, ingresando cualquier secuencia de nucleótidos (de alrededor de 20 nt de tamaño) y ver los resultados por pontalla. Esta aplicación está enfocada para que la utilicen biólogos (biotecnólogos, biólogos moleculares, etc), es por esto, que se ofrece una interfaz amigable para que estos usuarios puedan ver en detalle los resultados y seleccionar los genes de interés en forma específica y selectiva.

Para los microARNs conservados de la figura 3.9 guardamos los resultados en una base de datos para luego utilizar estos datos en una herramienta final llamada target3VU que se detalla en el capítulo 6. Esta herramienta fue desarrollada en Perl utilizando Perl DBI y MySQL para el almacenamiento y manejo de datos. Target3VU

utiliza datos almacenados en la base de datos que fueron generados al realizar la búsqueda de los microARNs conservados con la herramienta PatmatchMicro.

Estructura de la Base de Datos

Vamos a mostrar el tipo de datos que usamos mediante la definición de la creación de una tabla distinta para cada microARN conservado. Para cada microARN tenemos una tabla con los siguientes campos.

```
CREATE TABLE [IF NOT EXISTS] miR396(
    id INT(10) NOT NULL AUTO_INCREMENT,
    specie VARCHAR(50) NOT NULL,
    gen VARCHAR(30) NOT NULL,
    target VARCHAR(30) NOT NULL,
    align VARCHAR(30) NOT NULL,
    mirna VARCHAR(30) NOT NULL,
    mm INT(11) NOT NULL,
    ins INT(11) NOT NULL,
    del INT(11) NOT NULL,
    sust INT(11) NOT NULL,
    locusID VARCHAR(20),
    score INT(11),
    evalue FLOAT,
    mfe FLOAT NOT NULL,
    mm_filter INT(2),
    description INT(10),
    PRIMARY KEY(id)
);
```

Donde

- *id*: es un identificador único para cada gen encontrado.
- *specie*: es la especie que coincide con *Specie* de la figura 3.17.
- *gen*: es el nombre del gen que coincide con *Sequence_name* de la figura 3.17
- *target*: es el blanco corresponiente al par microARN-gen blanco que da como blanco al hacer la búsqueda de una secuencia de un microARN y coincide con el *target* de la figura 3.17.
- *align*: es el alineamiento corresponiente al par microARN-geb blanco que da como blanco al hacer la búsqueda de una secuencia de un microARN y coincide con el *Alignment* de la figura 3.17.
- *mirna*: es la secuencia del microARN de entrada de la búsqueda (en dirección 3'-5') y coincide con el *miRNA* de la figura 3.17.
- *mm*: es el número de *mismatches* de cualquier tipo que tiene el apareamiento del par microARN-gen blanco encontrado.
- *ins*: es el número de *mismatches* de tipo inserción que tiene el apareamiento del par microARN-gen blanco encontrado.

- *del*: es el número de *mismatches* de tipo deleción que tiene el apareamiento del par microARN-gen blanco encontrado.
- *sust*: es el número de *mismatches* de tipo sustituciónes que tiene el apareamiento del par microARN-gen blanco encontrado (en nuestro caso este valor es siempre igual a 0).
- *locusID*: es nombre del gen en Arabidopsis thaliana que aparece como mejor resultado del módulo Blast y coincide con con el *Best Hit* de la figura 3.17 (tiene el valor NULL en caso de que no se encuentre un buen candidato).
- *score*: es el puntaje que aparece como mejor resultado del módulo Blast y coincide con con el *Score* de la figura 3.17.
- *evalue*: es *Evalue* que aparece como mejor resultado del módulo Blast y coincide con con el *e-Value* de la figura 3.17.
- *mfe*: es la mínima energía de hibridación calculada con el módulo RNAHybrid del par microARN-gen blanco que coincide con mfe de la figura 3.17.
- *mm_filter*: Puede tener el valor 1 o 0 depende si pasa o no el filtro de posición de *mismatches* como se detalla en la sección 5.4.
- *Description*: Description family subfamily

3.4. Bases de datos y lenguaje

Por último, vamos a explicar la elección del lenguaje utilizado y daremos una descripción del sistema de base de datos que empleamos para almacenar y acceder a los datos relevantes en esta estrategia bioinformática de búsquedas de genes blancos para microARNs conservados en plantas.

3.4.1. Sobre el uso de Perl

Perl[9] es el lenguaje predominante en el ára de Bioinformática para búsquedas de patrones y análisis de secuencias, de hecho existe una colección de módulos open-source de Perl llamada BioPerl[10] que facilitan el desarrollo de scripts para aplicaciones de bioinformática [10]. Una de las cosas que hace Perl tan popular para abordar tareas biológicas está precisamente en su origen ecléctico, este lenguaje proviene del lenguaje de programación C y del Shell de Unix, este origen va a proveer justo la flexibilidad y las herramientas de programación necesarias para resolver problemas de contexto biológico. Perl es básicamente un lenguaje de programación tipo "scripting" que toma elementos de UNIX tales como sed, grep, awk, shell scripting y la dinámica y enfoque del

[9] http://www.perl.org/
[10] http://www.bioperl.org/wiki/Main_Page

lenguaje de programación C. La razón fundamental del uso de Perl en bioinformática es su orientación a la manipulación de caracteres. Como explicamos anteriormente el ADN, ARN y demás moléculas biológicas pueden ser reducidas a cadenas de caracteres en sus componentes fundamentales, caracteres comos A, C, G, T, U que equivalen a elementos constitutivos y distintivos de dichas moléculas. Estas secuencias de caracteres que caracterizan a las moléculas les proporcionan su identidad y unicidad.

Además Perl ofrece una interfaz para bases de datos llamada DBI (DataBase Interface). DBI, es un módulo (librería o conjunto de funciones) de Perl que se especializa en procesar bases de datos de diversas compañías, una de las principales ventajas de este modulo DBI es su portabilidad, es decir si se construye un sistemas de bases de datos en MYSQL, el mismo sistema Perl se puede usar para Oracle, Sysbase, etc, con muy pequeños cambios en el codigo original.

3.4.2. Sobre MySQL

Utilizamos MySQL[11] como sistema de gestión de bases de datos relacionales. Ya que se ofrece bajo licencia *GNU GPL* (Licencia Pública General de GNU) y además es un sistema multihilo y multiusuario. Utiliza el lenguaje de consulta estructurado SQL que es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas. Las operaciones que pueden realizarse sobre las tablas son las de lectura, escritura, borrado y actualización de registros, llamadas Select, Insert, Delete y Update respectivamente, en el estándar SQL.

[11] http://www.mysql.com/

Capítulo 4

Controles

Para validar la estrategia utilizada estadísticamente y hacer un análisis más preciso del número real de genes blancos encontrados y falsos positivos realizamos dos enfoques distintos. En el primero lo que hicimos fue generar, para cada microARN conservado, un número de secuencias al azar o *scramble* y con esas secuencias repetimos las búsquedas para encontrar genes blancos de la misma manera que hicimos con los microARNs verdadero. De esta manera se consideró a los microARNs como señal y las secuencias *scramble* como el ruido. En el segundo enfoque hicimos una búsqueda con microARNs que son específicos de *Arabidopsis thaliana* y sólo se encuentran en esta especie.

4.1. Técnica de *Scramble*

Para crear un conjunto de secuencias como control negativo, para representar lo que es esperado por azar, generamos 20 versiones *scramble* para cada microARN por medio de una permutación de sus nucleótidos. Lo hicimos manteniendo la composición de dinucleótidos [3] de manera que se mantenga la composición de las bases pero no el orden. Partimos cada secuencia de microARNs en fragmentos de dos nucleótidos después permutamos estas secuencias para generar lo que llamamos las secuencias *scramble*.

Realizamos un módulo para generar estas secuencias utilizando como entrada la secuencia del microARN ya partida en fragmentos de a dinucleótidos en un archivo. Lo hicimos de a un dinucleótido por línea de manera consecutiva (random.txt). Utilizamos la función *rand* de Perl que retorna un número al azar mayor igual a 0 y menor que el valor que se pasa como argumento. Se detalla a continuación la función encargada de generar las secuencias *scramble*.

```
open(LTN,"random.txt");
@reg=<LTN>;
close(LTN);

sub random{
    for $j (1..@reg) {
```

```
        do {
          $sol = int(rand (@reg));
        } while $used[$sol]++;
        $reg[$sol] =~ s/\n//;
        print $reg[$sol];
    }
}
```

Luego chequeamos que las secuencias generadas sean todas distintas entre sí y distintas al microARN.

Procedemos a realizar las búsquedas utilizando la misma metodología que con los microARNs conservados. Con estos resultados tomamos un promedio con su desviación estándar de las 20 secuencias *scramble* para cada microARN y lo comparamos con el microARN real. Luego, con estos datos, comparamos el números de blancos predichos por nuestra estrategia para microARNs reales (señal) y el promedio predicho para microARNs *scramble*(ruido).

4.2. Técnica de microARNs no conservados

Como nuestra estrategia se basa en conservación evolutiva del sitio blanco vamos a usar un segundo control tomando dos microARNs que se saben que no están conservados en plantas [2]. Estos son el miR158 y el miR173 que están presentes en *Arabidopsis thaliana* pero no lo están en otras especies como *Oryza sativa* (arroz). Lo que hicimos fue realizar la búsqueda del mismo modo que lo hicimos para los microARNs conservados. Debido a que estos microARNs no estan conservados entonces el par microARN-gen blanco tampoco lo va a estar ya que las secuencias de nucleótidos sufren mutaciones (alteraciones en la información genética) al no ser que haya una presión de selección para que esto no ocurra.

Capítulo 5

Resultados y discusión

En este capítulo presentamos primero los distintos filtros (tanto empíricos como de conservación evolutiva) utilizados para la búsqeuda de genes blancos. También mostramos los resultados obtenidos con esta estrategia para los microARNs y sus secuencias *scramble*. Además mostramos que es lo que sucede con los microARNs control.

5.1. Búsquedas de potenciales genes blancos

Como se mencionó anteriormente los microARNs tienen secuencias de 21nt. Sin embargo, un análisis en la base de datos de microARN reveló que solo 18 de los 21 nt están conservados. Esto es debido a que existe variación en el nucleótido primero, veinte y veintiuno. Por lo tanto, consideramos sólo estos 18 nucleótidos para hacer la búsqueda lo cual simplifica la cantidad de genes a analizar. Tomamos cada secuencia de 18 nt correspondientes a los 22 microARNs conservados en plantas y procedimos a buscar sus potenciales blancos. En una primera búsqueda consideramos como potencial blanco a aquel gen que forme un par microARN-gen en al menos 15 de los 18 nt. Para esto hicimos una búsqueda con el modulo PatmatchMicro con la secuencia complementaria al microARN de 18 nt permitiendo tres mismatches.

5.2. Filtro mfe

A continuación calculamos la energía mínima de hibridación (mfe) que existe entre el microARN y la secuencia blanco utilizando el módulo implementado llamado RNAHybrid. Ya que además del alineamiento, esta energía de hibridación es importante porque nos da una idea de la magnitud con que el microARN y el gen blanco están interaccionando y cuanta energía se necesita para romper esa interacción.

Más grande en valor absoluto es la energía, más fuerte es la interacción. Por lo tanto consideramos un corte teniendo en cuenta esta propiedad biológica de la interacción microARN-gen blanco. Calculamos el 72 % de la energía perfecta de hibridación entre cada microARN conservado y su secuencia complementaria 72 %PE (el 72 % de la energía perfecta) y luego utilizamos este valor para hacer el corte. Claramente estos

valores de corte dependen de la secuencia de cada microARN en particular. Practicamente todos los pares microARN-gen blanco validados experimentalmente pasan este filtro [5].

El módulo RNAhybrid está integrado al programa PatmatchMicro. Debido a esto cuando realizamos las búsquedas de genes blancos para cada microARN conservado el valor de la mfe entre el par microARN-gen blanco es asignado y guardado en un campo de la tabla de la base de datos. Para luego comparar este valor guardado con el valor de corte que describimos anteriormente (72 %PE).

De esta manera nos quedamos con los genes blancos que tienen una mfe menor o igual al corte. Es decir filtramos algunos genes blancos donde su valor correspondiente a la mfe es mayor al corte. A este filtro lo llamamos filtro mfe.

5.3. Filtro mm

Empiricamente sabemos que no puede existir más de 1 *mismatch* de la posición 2 a la 12 del microARN [5] (para nuestro estudio tomamos de la 1 a la 11 debido a que eliminamos el primer nucleótido). Ya que se sabe que esta no es una interacción permitida porque da como resultado un microARN no funcional.

Agregamos esto por medio de otro filtro. Para cada interacción agregamos una nueva columna en la base de datos que representa la condición de ese alineamiento en particular, es decir si es una interacción permitida o no. Luego nos quedamos, mediante consulta a la base de datos, solamente los que pasan este filtro de posición de *mismatches*.

En general cuando mostramos un alineamiento, el microARN lo colocamos en dirección inversa ($miR = \alpha_{18}\alpha_{17}...\alpha_1$), es decir 3'-5'. Y la secuencia blanco la mostramos en dirección original, es decir 5'-3'. De esta manera decidimos si el alineamiento está permitido chequeando que no haya más de 1 *mismatch* de la posición 1 a la posición 11 como se muestra en la figura 5.1.

Figura 5.1: Interacción permitida y no permitida

5.4. Filtros empíricos

Tanto el filtro de energía (filtro mfe) y el filtro de posición de *mismatches* (filtro mm) son reglas empíricas que fueron tenidas en cuenta a partir de experimentación biológica

en trabajos anteriores. Sin embargo estas reglas no son absolutas y provocan que la búsqueda de blancos sea bastante selectiva, perdiendo un gran número de blancos reales pero reduciendo también los falsos positivos. Perdiendo sensibilidad en la estrategia utilizada pero ganando especificidad.

Tomamos los 22 microARNs conservados utilizados en este trabajo y comparamos el número de genes blancos (sin aplicar ningún filtro) individualmente para cada microARN y el promedio de sus secuencias *scramble*. Luego repetimos esto, pero ahora aplicando filtro de energía (mfe) y comparamos los resultados obtenidos. Y por último a los resultados obtenidos, le aplicamos el filtro de posición de *mismatches* (mm) de manera que ambos filtros actúen conjuntamente. A modo de ejemplo, mostramos en la figura 5.2 lo que sucede con la relación señal/ruido (microARNs/*scramble*) al ir aplicando los distintos filtros mencionados anteriormente para los genes obtenidos con el total de 22 microARNs conservados.

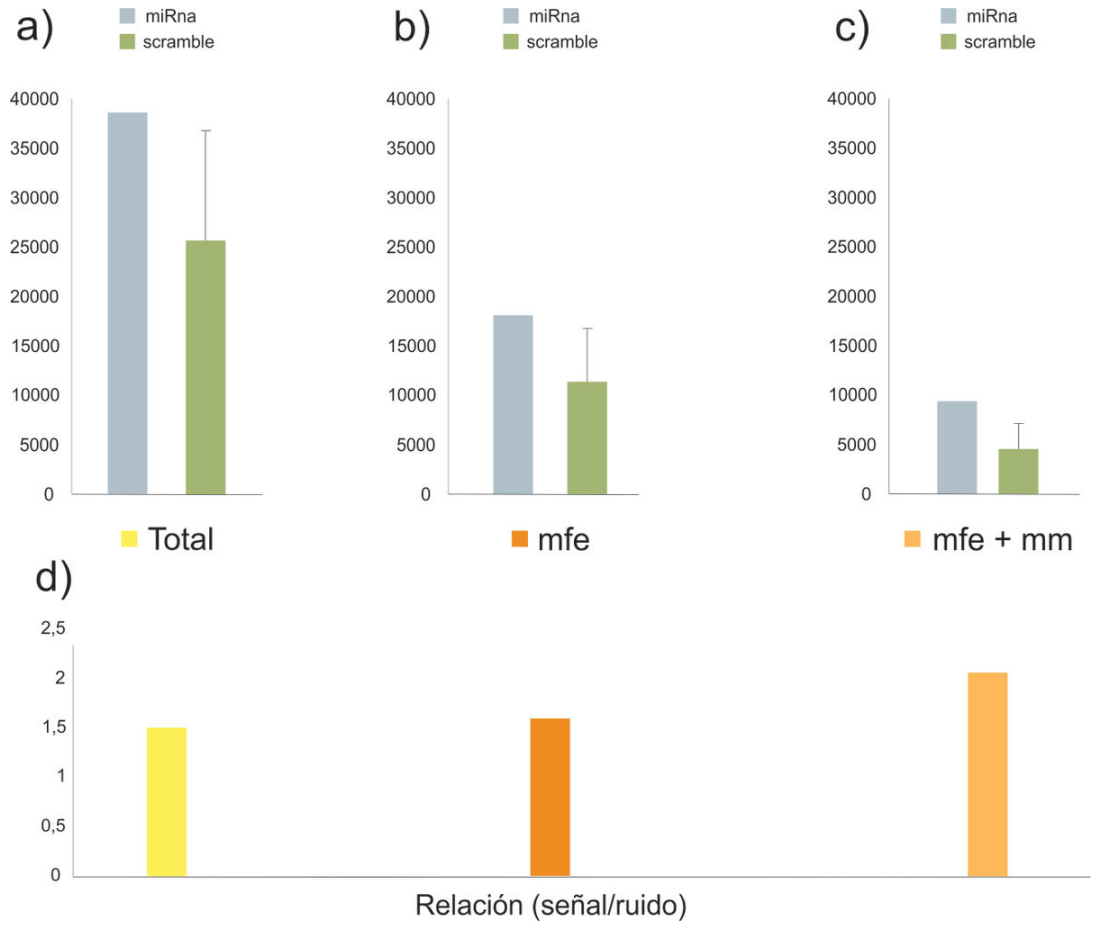

Figura 5.2: a) Número total microARNs vs *Scramble* sin aplicar ningún filtro. b) Número total microARNs vs *Scramble* aplicando filtro de mfe. c) Número total microARNs vs *Scramble* aplicando filtro de mfe y de mm. d) Relación señal/ruido para a), b) y c)

Como podemos ver en la figura 5.2 a) el número total de genes blancos que aparecen para todos los microARNs es similar a la suma del promedio de las 20 secuencias *scramble* para cada microARN. Es decir que para la búsqueda inicial (aparecen 38.597 genes blancos para los 22 microARNs) el número total es realivamente cercano a lo que ocurre por azar (teniendo en cuenta su desviación estándar).

Luego vemos en la figura 5.2 b) que al aplicar el filtro de mfe el número de genes para los microARNs baja considerablemente pero también ocurre esto (en mayor proporción) para las secuencias *scramble*, mejorando así la relación señal/ruido.

A continuación, en la figura 5.2 c) podemos ver que al aplicar ambos filtros (el de mfe y el de mm) el número de genes blancos para los microARNs baja aún mas y sucede lo mismo con el promedio de las secuencias *scramble*, descartando de esta manera un gran número de falsos positivos.

Y por último en la 5.2 d) vemos como incrementa la relación señal/ruido al aplicar sucesivamente los distintos filtros, pasando de una relación de 1,5 hasta llegar a una relación del doble. Sugiriendo además que los filtros de mfe y de mm, al aplicarlos conjuntamente, producen un resultado aditivo sobre la búsqueda de genes blancos. Sin embargo, puede observarse que la relación señal/ruido está alrededor de 2, lo cual indica que hay muchos falsos positivos en esta búsqueda.

5.5. Evolución

En trabajos anteriores se habían hecho estudios similares aplicando los filtros mencionados en la sección anterior. Pero en este trabajo proponemos además una regla nueva, considerando la conservación evolutiva del par microARN-gen blanco.

En el momento de hacer la búsqueda y por medio del módulo Blast integrado en la herramienta PatmatchMicro a cada gen encontrado le asignamos un homólogo en *Arabidopsis thaliana* identificandolo con su LocusID. De este modo todos los genes (no pertenecientes a Arabidopsis) que aparecen como blanco para algún microARN tienen asignado un valor en la columna LocusID de la base de datos. Los valores de LocusID para los genes que aparecen en *Arabidopsis thaliana* coinciden con el valor correspondiente al nombre del gen (SequenceName).

Agrupamos los genes por el LocusID (identificador) de Arabidopsis thaliana que fueron asignados para cada gen por medio del módulo Blast. Luego pedimos conservación evolutiva en al menos 4 especies como muestra la figura 5.3 a). Es decir que para que un gen en Arabidopsis thaliana, identificado por ejemplo con el LocusID AT1G23230, aparezca dentro de los candidatos tiene que existir al menos 4 genes de distintas especies que tengan asignado con el módulo Blast el LocusID igual a AT1G23230.

Luego buscamos conservación en al menos 5 especies como muestra la figura 5.3 b) y comparamos ambos resultados para ver el número de genes que aparecen en ambas estrategias y ver la relación señal/ruido, como mostramos en la figura 5.3 c).

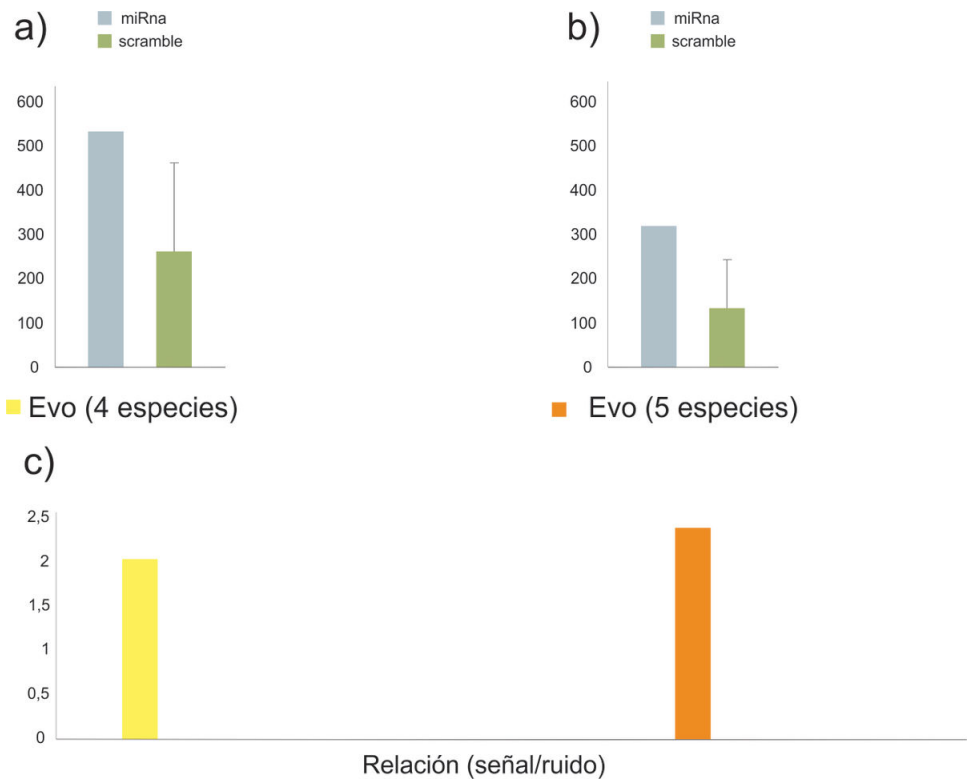

Figura 5.3: a) Número genes blancos que se conservan en al menos 4 especies para microARNs vs *Scramble*. b) Número genes blancos que se conservan en al menos 5 especies para microARNs vs *Scramble*. c) Relación señal/ruido para a) y b)

5.6. Filtros empíricos y evolución

Ahora lo que hicimos fue aplicar el filtro de conservación evolutiva (evo) sobre ambos filtros empíricos (mfe y mm), como se muestra en la figura 5.4. Lo hicimos buscando conservación evolutiva en al menos 4 especies (figura 5.4 a) y aplicando filtro de mm y de evolución y buscando conservación evolutiva en al menos 5 especies (figura 5.4 b). Lo que vemos que estos filtros actúan de manera sinérgica, es decir que el filtro de evolución actúa por separado con los filtros de mm y de mfe, pero ambos cooperando en la relación señal/ruido y en la selección de genes blancos. Viendo así que, tanto para 4 especies como para 5 especies, la relación incrementa notablemente pasando de una relación de 1,5 cuando no aplicábamos ningún filtro hasta llegar a una relación de 7,3 al aplicar todos los filtros. Reduciendo por medio de la estrategia el número de genes blancos encontrados hasta quedarnos selectivamente con 114 (en el caso de conservación en 5 especies).

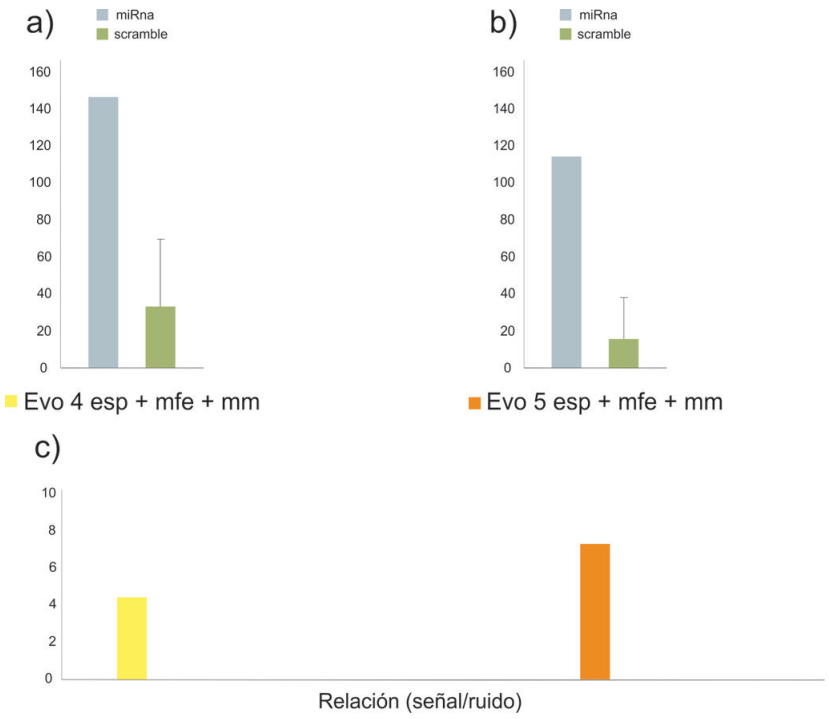

Figura 5.4: a) Número genes blancos que se conservan en al menos 4 especies con filtro de mfe y de mm para microARNs vs *Scramble*. b) Número genes blancos que se conservan en al menos 5 especies con filtro de mfe y de mm para microARNs vs *Scramble*. c) Relación señal/ruido para a) y b)

5.7. Controles

Como control y a modo de ejemplo, tomamos en el miR158 que sabemos que este microARN no está conservado a lo largo de las especies que estamos estudiamos, pero si está presente en Arabidosis thaliana. Es por esto que lo utilizamos para validar nuestra estrategia en la parte de conservación evolutiva (evo). Vemos en la figura 5.5 a,b y c que al aplicar sucesivamente los filtros de mfe y de mm el número de genes blancos para el miR158 se reduce en similar magnitud que el promedio de sus secuencias *scramble*. Y en la figura 5.5 d vemos que la relación señal/ruido se mantiene constante indicándonos que el filtro no estaría funcionando para este microARN no conservado.

Luego vemos que sucede con el filtro de evolución al buscar conservación en al menos 4 especies como mostramos en la figura 5.6 a). Lo que vemos aquí es que al buscar genes blancos utilizando conservación evolutiva, el número de genes que aparece para el miR158 (15 genes) es significativamente menor al número de genes para los microARNs conservados (25 en promedio) y similar (e incluso menor) al promedio de sus secuencias *scramble*. En la figura 5.6 b vemos que al aplicar el filtro evo sobre los filtros de mfe y mm, el número de genes blancos para el mir158 cae notablemente. De manera que se encuentra 1 sólo gen blanco para el mir158 y 2 en promedio para sus

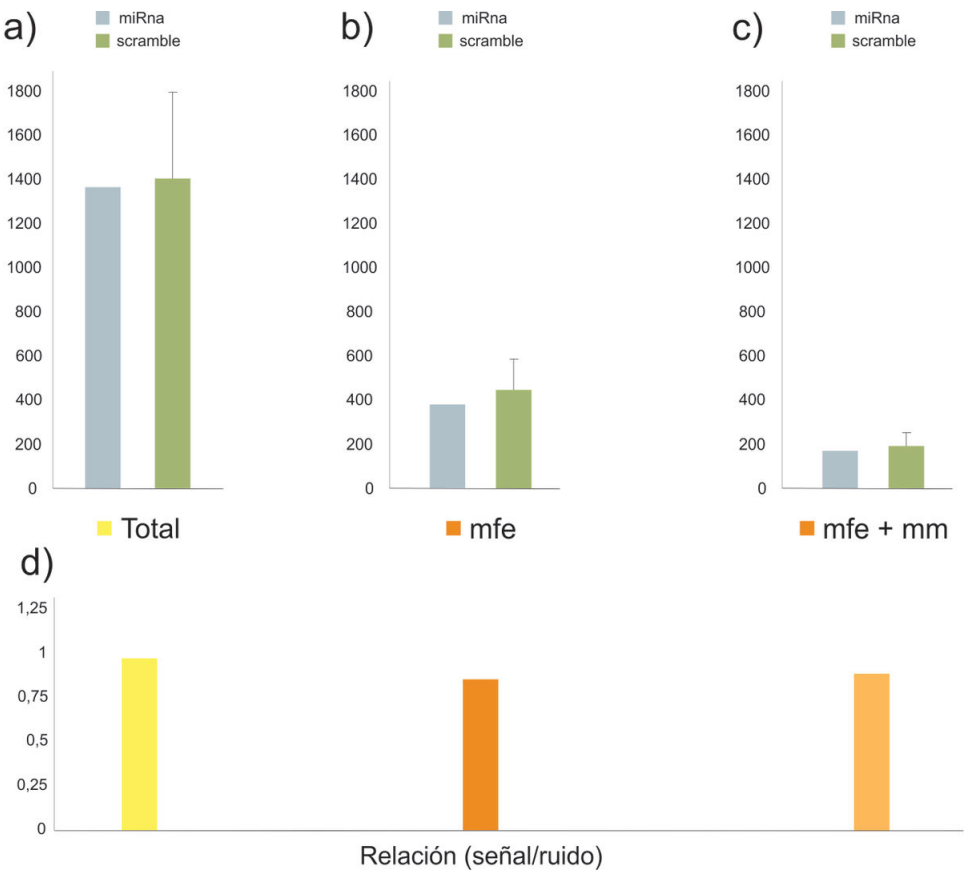

Figura 5.5: Control de microARN no conservado: miR158. Aplicando sucesivamente los filtros de mfe y mm.

secuencias *scramble*. Lo que nos estaría diciendo que la estrategia elimina casi en su totalidad a los genes blancos encontrados en la búsqueda y esto valida nuestra teoría donde esperábamos que la estrategia no sirva para microARNs no conservados.

Por último en la figura 5.6 c) mostramos la relación señal/ruido donde vemos que para este microARN no conservado la relación es siempre similar a 1.

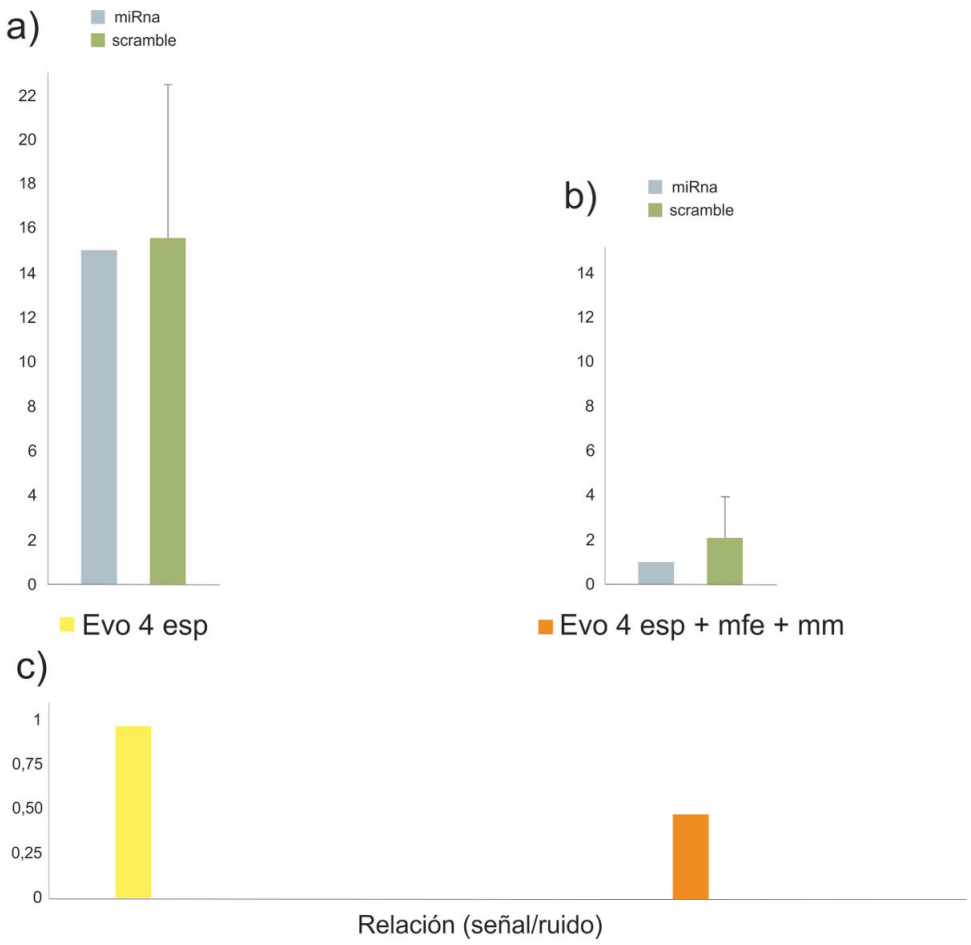

Figura 5.6: a) Número de genes blancos que se conservan en al menos 4 especies para el miR158 vs *Scramble*. b) Número de genes blancos que se conservan en al menos 4 especies con filtro de mfe y de mm para el miR158 vs *Scramble*. c) Relación señal/ruido para a) y b)

Capítulo 6

Target3VU

Este es un trabajo interdisciplinario en donde se desarrolló una estrategia y una herramienta para resolver un problema de índole biológico que sobrepasa el discernimiento humano. Es por esto que se presenta en este capítulo, una herramienta final en la cuál biólogos moleculares (colegas del laboratorio), se basaron para seleccionar nuevos potenciales genes blancos de microARNs y los validaron experimentalmente en el laboratorio.

Esta validación experimental se realizó en colaboración con estudiantes del Doctorado en Ciencias Biológicas del laboratorio donde se desarrolló esta tesina, por medio de la téecnica 5'-RACE PCR modificada [20].

6.1. Interfaz

Bajo plataforma Linux, con un entorno web desarrollado en Perl DBI y utilizando Mysql para almacenar y acceder a los datos, desarrollamos una herramienta llamada Target3VU como se muestra en la figura 6.1. Esta figura representa la segunda parte de nuestra estrategia de búsquedas de genes blancos que se mostró anteriormente en la figura 3.1.

Esta herramienta web fue desarrollada en principio para estudiantes del Doctorado y Postdoctorado en Ciencias Biológicas del laboratorio donde se desarrolló esta Tesis. El programa fue montado en un servidor web con Apache en linux de manera que los usuarios puedan acceder a la herramienta a través de una Intranet mediante cualquier navegador. En un futuro se planea ofrecer el acceso libre a la herramienta y los datos para la comunidad científica bajo licencia GNU.

A continuación detallaremos la heramienta final Target3VU. Para los 22 microARNs conservados que se utilizaron en este trabajo, la búsqueda de genes blancos que se llevo a cabo, se almacenó en una base de datos. De esta manera en la entrada de la herramienta que se muestra en la figura 6.2 el usuario puede elegir el microARN identificandolo por su número. Luego se permite (por medio de un checkbox) la opción de utilizar el filtro de posición de *mismatches* (mm) y también elegir el corte del filtro de energía de hibridación (mfe). Y por último, teniendo el cuenta el filtro de conservación evolutiva

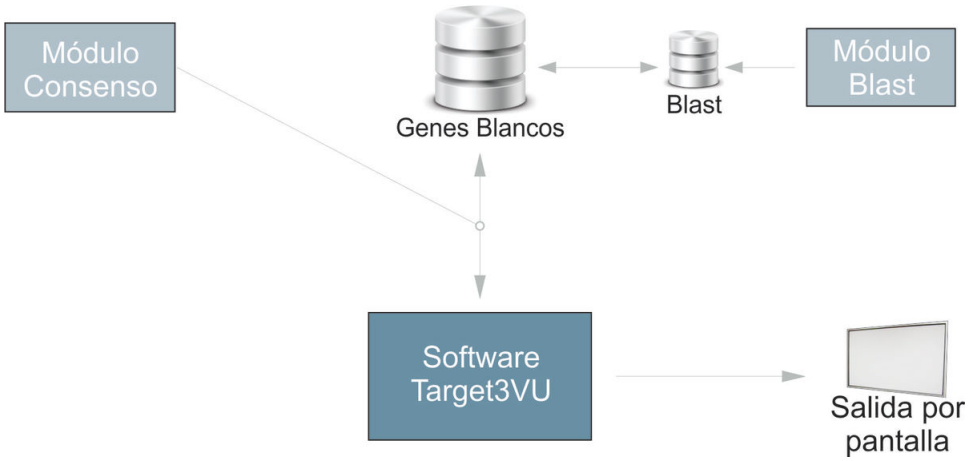

Figura 6.1: Arquitectura de la herramienta Target3VU

(evo), el usuario puede elegir el número mínimo de especies en donde se busca que la secuencia blanco esté presente. En la figura 6.2 los parámetros de búsqueda estan configurados para hacer la búsuqueda del microARN miR396 con filtro de mm y corte de mfe del 72 % de la energía perfecta y además se busca conservación en al menos 4 especies.

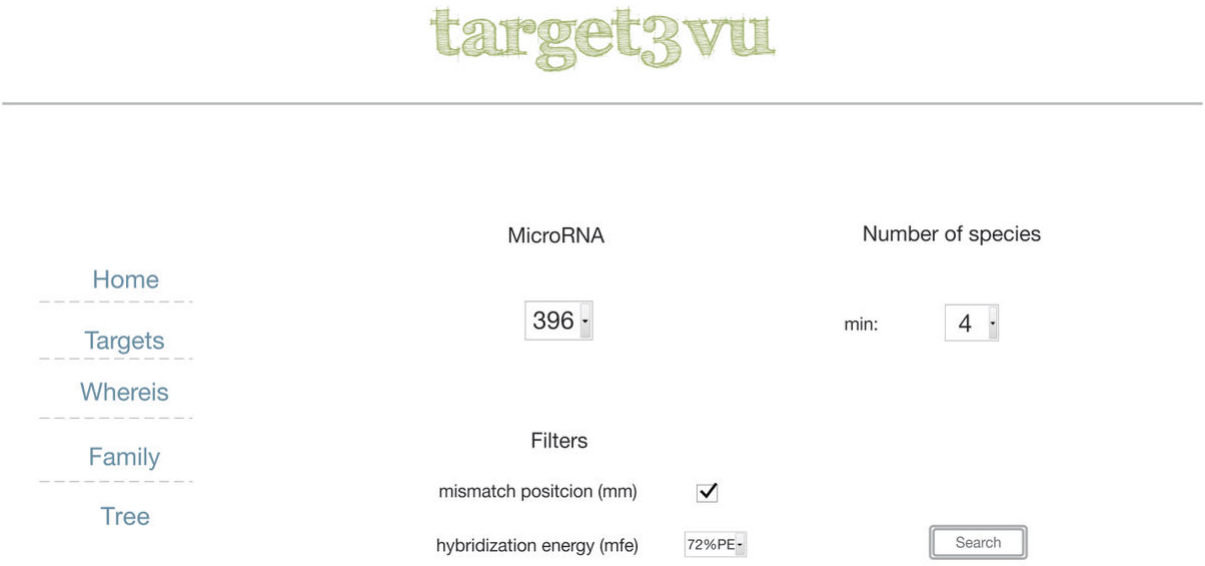

Figura 6.2: Pantalla de inicio de la herramienta Target3VU.

Como resultado de buscar genes blancos para un microARN en particular mostramos la figura 6.3 donde se refleja parte del resultado de la búsqueda de la figura 6.2. Se muestra el identificador de secuencia (locusID) de *Arabidopsis thaliana*. Este locusID es el que tienen en común todos los genes blancos que aparecen en las distintas especies

en la columna *Species* asignados mediante el módulo Blast. Luego aparece una pequeña descripción funcional del gen blanco identificado con el locusID, así como también la familia a la que pertenece. Además el usuario puede ver la secuencia completa haciendo click sobre el locusID y ver los distintos alineamientos del par microARN-gen blanco en las distintas especies como se muestra en la figura 6.4. Esto permite al usuario saber si la interacción microARN-gen blanco esta conservada.

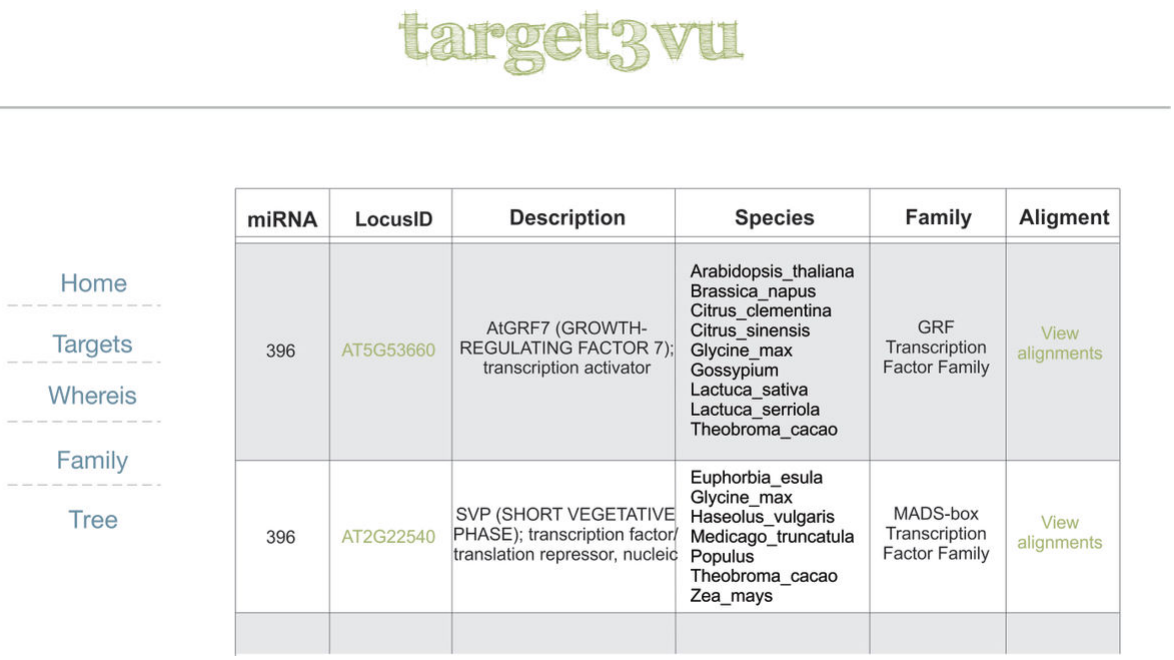

Figura 6.3: Resultado de buscar genes blancos para un microARN con la herramienta Target3VU

En esta figura 6.4, se muestra (entre paréntesis en la celda que dice mfe) el corte que se utilzó para hacer la búsqueda del filtro de mfe. Y luego para cada resultado se marca en color rojo, si un determinado gen blanco de una determinida especie, no pasó este filtro. Lo mismo sucede con el alineamiento, donde se marca si el gen no tiene un alineamiento permitido (teniendo en cuentra el filtro de mm). Además se muestra el nombre del gen y de la especie del gen blanco encontrado.

La herramienta target3VU ofrece otras opciones y posibilidades de búsqueda. A la izquierda, en el menu de la figura 6.2 y 6.3 , se puede ver que además de la búsqueda de genes blancos (por medio de la opción de menu: *Targets*) se ofrece otras opciones como *Whereis*. Esta opción permite hacer búsquedas a la inversa. Es decir, el usuario ingresa el nombre de una secuencia identificada con el locusID de *Arabidopsis thaliana* y la herramienta busca los microARNs donde aparece esta secuencia como blanco. Esto puede ser útil cuando un mismo gen es regulado por más de un microARN.

En la opción *Family* los parámetros de entrada son los mismos que para *Targets* la diferencia es que el resultado en vez de agrupar genes blancos por locusID en este

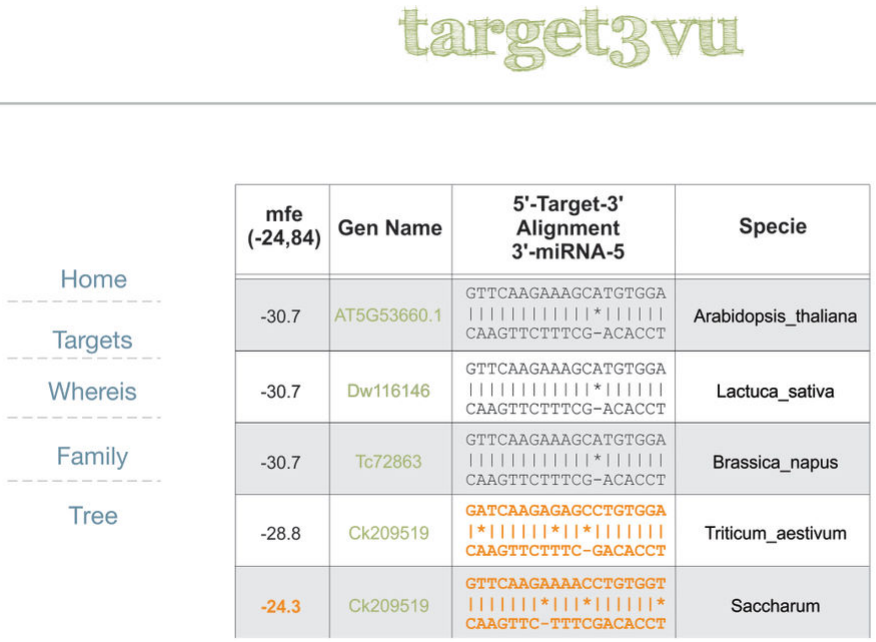

Figura 6.4: Distintos alineamientos de la búsqueda de genes blancos para un microARN utilizando la herramienta Target3VU

caso los genes blancos se agrupan por familia (si es que se conoce). Entonces se puede tener varios genes identificados por su locusID perteneciente a una misma familia. Esto puede ser útil debido a que varios genes similares pertenecientes a una misma familia pueden ser regulados por un mismo microARN, de esta manera se pueden encontrar genes que se omitieron en la búsqueda convencional debido a falta de información en los genomas de las plantas.

Y por último en la opción *Tree* se puede ver en un árbol filogenético marcadas en color las especies en donde aparece un determinado gen blanco. Un árbol filogenético es un árbol que muestra las relaciones evolutivas entre varias especies que tienen una ascendencia común. Entonces en la búsqueda de genes blancos, para cada gen blanco como resultado el usuario puede ver si dicho gen esta presente en especies cercanas o si está distribuido a lo largo de las especies. Esto es útil, por ejemplo, cuando se quiere saber en que parte de la evolución de las especies aparece un gen como blanco de un microARN. Esta opción actualmente no esta integrada en la herramienta, ya que en un principio la estrategia se había hecho con bases de datos de plantas tomadas de *J. Craig Venter Institute*[1] y esta opción si estaba disponible, pero luego se optó por las bases de datos usadas actualmente ya que además de no ser redundantes, están actualizadas. Y la clasificación de estas nuevas especies en un árbol filogenético no lo hemos realizado, así como tampoco el módulo que integra esta opción del árbol para estas nuevas bases de datos.

[1]http://www.jcvi.org/

6.2. Validación experimental

Con esta nueva herramienta que detallamos en el capítulo 6, colegas del laboratorio eligieron los mejores candidatos para luego validarlos experimentalmente. Los mejores candidatos se encontraron utilizando la estrategia que generó la mejor señal/ruido detalladas en el cápítulo 5. A partir de aquí se eligieron diez genes y se pudieron validar con éxito 5 genes nuevos. Desde un punto de vista biológico este resultado es importante.

Además de los genes blancos nuevos validados, con esta herramienta se encontraron un gran número de genes blancos que podrían ser estudiados posteriormente y genes blancos validados en trabajos anteriores como se muestra en la figura 6.5. Luego para buscar otros candidatos se utilzaron distintos enfoques relajando la búsqueda de genes blancos y utilizando las distintas opciones de búsqueda que la herramienta Target3VU ofrece.

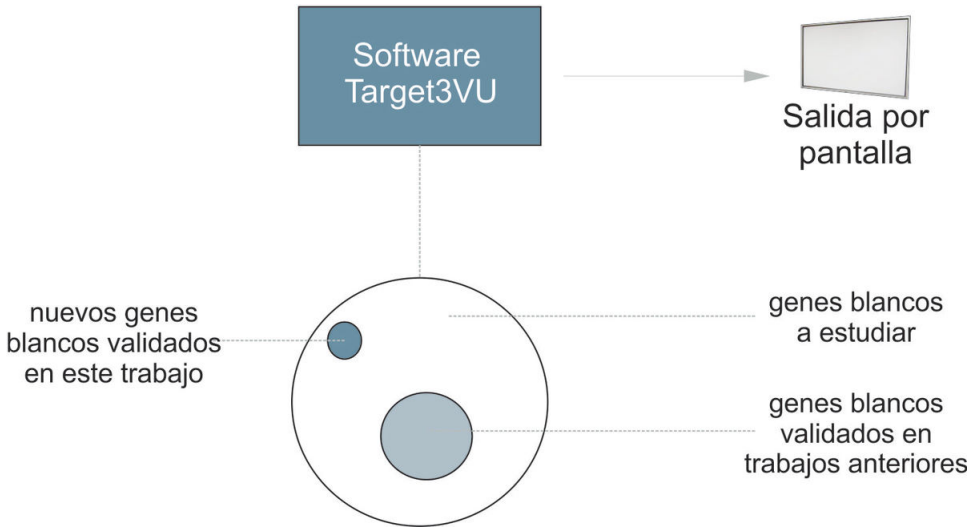

Figura 6.5: Conjunto de genes blancos que da como resultado de hacer la búsqueda de genes blancos para microARNs conservados con la herramienta Target3VU.

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

La identificación de genes es uno de los primeros y más importantes pasos para entender el genoma de una especie una vez que ha sido secuenciado. Este trabajo brinda una nueva metodología para la identificacion de genes regulados por microARNs. La estrategia utilizada puede llegar a descartar genes blancos reales. Sin embargo la idea de este enfoque no es encontrar todos, sino tratar de encontrar nuevos genes blancos con función biológica, teniendo un número pequeño de falsos positivos (genes que no son blanco de ninguno de los microARNs conservados). Este último punto es esencial porque el número de falsos positivos genera una gran complejidad a la hora de realizar experimentos biológicos en el laboratorio por los colegas con formación en Biología Molecular. A modo de resumen se detallan algunos de los puntos importantes en este trabajo.

Logramos encontrar y validar experimentalmente nuevos genes blancos de nuevas familias reguladas por microARNs.

Logramos plantear una estrategia y desarrollar nuevas herramientas para la identificación de genes blancos en plantas. Ofreciendo un nuevo enfoque, centrando la estrategia, en la conservación evolutiva del sitio blanco.

Logramos adaptar un software general, utilizado para nucleótidos y proteínas, para realizar búsquedas particulares de genes blancos de microARNs conservados.

Desarrollamos un nuevo software a partir de datos almacenados de genes blancos de microARNs para que estudiantes del Doctorado y Posdoctorado en Ciencias Biológicas lo utilicen para encontrar nuevos genes blancos. Y en un futuro planeamos ofrecer el acceso libre a la herramienta y los datos para la comunidad científica bajo licencia GNU.

7.2. Trabajos futuros

Trabajos futuros en esta línea pueden brindar mejoras a esta estrategia y nuevas metodologías que derivan de la propuesta, como las siguientes:

- Nuevos enfoques de búsquedas de genes blancos en plantas basados en conservacón evolutiva, tratando de lograr mayor sensibilidad o especificidad agregando nuevos parámetros o modificando los implementados en este trabajo.

- Búsqueda de genes blancos regulados por más de un microARN o búsquedas de genes blancos con interacciones particulares poco comunes.

- Identificar genes blancos regulados por microARNs que aparecieron en algún momento particular de la evolución de la especies de plantas consideradas.

- Mejora del algoritmo nrgrep para separar los diferentes tipos de mismatches, permitiendo hacer la búsqueda con tipo y número de mismatches independientes.

Bibliografía

[1] Schauer S E Jacobsen S E Meinke D W Ray A. Dicer-like1: blind men and elephants in arabidopsis development. *Trends Plant Sci*, 2002.

[2] Jones-Rhoades M W Bartel D P Bartel B. Micrornas and their regulatory roles in plants. *Rev. Plant Biol*, 2006.

[3] Lewis BP Shih IH Jones-Rhoades MW Bartel DP Burge CB. Prediction of mammalian microrna targets. *Cell*, 2003.

[4] Palatnik J F Allen E Wu X Schommer C Schwab R Carrington J C Weigel D. Control of leaf morphogenesis by micrornas. *Nature*, 2003.

[5] Schwab R Palatnik JF Riester M Schommer C Schmid M Weigel D. Specific effects of micrornas on the plant transcriptome. *Developmental Cell*, 2005.

[6] Huang JY Brutlag DL. tagca grep for dna. *BMC Bioinformatics*, 2002.

[7] Reinhart BJ Weinstein EG Rhoades MW Bartel B Bartel DP. Micrornas in plants. *Genes Dev*, 2002.

[8] Rhoades MW Reinhart BJ Lim LP Burge CB Bartel B Bartel DP. Prediction of plant microrna targets. *Cell*, 2002.

[9] Yekta S Shih IH Bartel DP. Microrna-directed cleavage of hoxb8 mrna. *Science*, 2004.

[10] Stajich JE Block D Boulez K Brenner SE Chervitz SA Dagdigian C Fuellen G Gilbert JG Korf I Lapp H Lehväslaiho H Matsalla C Mungall CJ Osborne BI Pocock MR Schattner P Senger M Stein LD Stupka E Wilkinson MD Birney E. The bioperl toolkit: Perl modules for the life sciences. *Genome Res*, 2002.

[11] Mraz M Pospisilova S Malinova K et al. Micrornas in chronic lymphocytic leukemia pathogenesis and disease subtypes. *Leuk Lymphoma*, 2009.

[12] Zhao Y Ransom J F Li A et al. Dysregulation of cardiogenesis, cardiac conduction, and cell cycle in mice lacking mirna-1-2. *Cell*, 2007.

[13] Crick F. Central dogma of molecular biology. *Nature*, 1970.

[14] Baeza-Yates R Gonnet G. A new approach to text searching. *Communications of the ACM*, 1992.

[15] Grillo G Licciulli F Liuni S Sbisa E Pesole G. Patsearch: a program for the detection of patterns and structural motifs in nucleotide sequence. *Nucleic acid research*, 2003.

[16] Navarro G. A guided tour to approximate string matching. *ACM Computing Surveys*, 2001.

[17] Navarro G. NR-grep: a fast and flexible pattern matching tool. *Software Practice and Experience (SPE)*, 31:1265–1312, 2001.

[18] Mangalam HJ. The emotif database. *Nucleic Acids Res.*, 2001.

[19] Fahlgren N Jogdeo S Kasschau KD Sullivan CM Chapman EJ Laubinger S Smith LM Dasenko M Givan SA Weigel D Carrington JC. Microrna gene evolution in arabidopsis lyrata and arabidopsis thaliana. *Plant Cell*, 2010.

[20] Bologna NG Mateos JL Bresso EG Palatnik JF. A loop-to-base processing mechanism underlies the biogenesis of plant micrornas mir319 and mir159. *EMBO J*, 2009.

[21] Thompson K. Regular expression search algorithm. *Communications of the ACM*, 1968.

[22] Navarro G Raffinot M. A bit-parallel approach to suffix automata: Fast extended string matching. In *Proc. 9th Annual Symposium on Combinatorial Pattern Matching (CPM)*, LNCS v. 1448, pages 14–33. Springer-Verlag, 1998.

[23] Navarro G Raffinot M. Fast and flexible string matching by combining bit-parallelism and suffix automata. *ACM Journal of Experimental Algorithmics (JEA)*, 2000.

[24] Pesole G Liuni S D'Souza M. Spatsearch: a pattern matcher software that finds functional elements in nucleotide and protein sequences and assesses their statistical significance. *Bioinformatics*, 2000.

[25] G. Navarro and M. Raffinot. Fast regular expression search. In *Proc. 3rd Workshop on Algorithm Engineering (WAE)*, LNCS 1668, pages 198–212, 1999.

[26] G. Navarro and M. Raffinot. Compact dfa representation for fast regular expression search. In *Proc. 5th Workshop on Algorithm Engineering (WAE)*, LNCS 2141, pages 1–12, 2001.

[27] Jones-Rhoades M W Bartel D P. Computational identification of plant micrornas and their targets, including a stress-induced mirna. *Mol Cell*, 2004.

[28] Berry G Sethi R. From regular expression to deterministic automatoa. *Theorical Computer Science*, 1986.

[29] Rehmsmeier M Steffen P Höchsmann M Giegerich R. Fast and effective prediction of microrna/target duplexes. *RNA*, 2004.

[30] Yan T Yoo D Berardini TZ Mueller LA Weems DC Weng S Cherry JM Rhee SY. Patmatch: a program for finding patterns in peptide and nucleotide sequences. *Nucleic Acids Res*, 2005.

[31] Alves Junior L Niemeier S Haunschild Arne Rehmsmeier M Merkle T. Comprehensive prediction of novel microrna targets in arabidopsis thaliana. *Nucleic Acids Research*, 2009.

[32] Wu S Manber U. Fast text searching allowing errors. *Communications of the ACM*, 1992.

[33] Wienholds E Koudijs M J van Eeden F J Cuppen E Plasterk R H. The microrna producing-enzyme dicer1 is essential for zebrafish development. *Nat Genet*, 2003.

[34] Crochemore M Rytter W. Text algorithms. *Oxford University Press*, 1994.

[35] Czumaj A Crochemore M Gasieniec L Jarominek S Lecroq T Plandowski W Rytter W. Speeding up two string-matching algorithms. *Algorithmica*, 1994.

[36] Baeza yates R. Text retrieval: Theory and practice. In *In 12th IFIP World Computer Congress, volume I*, pages 465–476. Elsevier Science, 1992.