UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE CIENCIAS EXACTAS Y NATURALES

DEPARTAMENTO DE COMPUTACIÓN

Búsqueda de motivos funcionales en proteínas mediante una heurística Greedy utilizando la Teoría de la Información Molecular

Functional motif search in proteins by a Greedy heuristic
using Molecular Information Theory

Tesis presentada para optar al título de
Licenciado en Ciencias de la computación

Leandro Radusky - L.U. 007/05

Director: Dr. Ignacio Enrique Sánchez

Buenos Aires, 2011

Resumen

El presente trabajo se propone implementar un algoritmo capaz de identificar y describir de forma cuantitativa el motivo lineal cuya presencia hace posible una interacción entre dos proteínas.
El problema biológico que se pretende abordar es la identificación de motivos lineales de proteínas responsables del establecimiento de interacciones físicas con otras proteínas. En los últimos años, los métodos experimentales de alto rendimiento han permitido la identificación de un gran número de interacciones físicas entre proteínas. Tales métodos proporcionan la identidad de las proteínas interactuantes, pero ningún detalle de las interacciones moleculares que hacen posible esta interacción.
Desde el punto de vista computacional, se conocen las secuencias que forman un conjunto de proteínas interactuantes (las cuales son representadas mediante cadenas de caracteres, cada uno de estos caracteres representando una molécula constitutiva de la proteína, denominada monómero) y se busca el patrón de caracteres con mayor sobrerrepresentación en el conjunto de entrada. Denominamos “motivo lineal” a este patrón.
Consideramos a este patrón o motivo sobrerrepresentado como una característica compartida por las secuencias de caracteres (representando proteínas) de entrada del algoritmo, y por lo tanto candidato a ser la secuencia de monómeros que permiten la interacción de estas proteínas con una misma proteína, denominada “compañero de interacción”.
Se implementa para los motivos lineales una forma de visualización probabilística, la cual permite conocer la certidumbre de encontrar un determinado monómero en una posición particular del motivo.
Multitud de interacciones entre proteínas dependen fuertemente de uno o varios motivos lineales. Los motivos lineales de proteínas consisten en pequeños fragmentos de la secuencia de una proteína, típicamente de entre 5 y 15 monómeros de longitud total.
Trabajos publicados por los grupos de Thomas Schneider y Gary Stormo muestran que la teoría de la información de Shannon puede adaptarse al estudio de las interacciones moleculares entre proteínas y motivos lineales de ADN. El resultado de dicha adaptación se denomina “Teoría de la Información Molecular”.
Esta teoría nos permite puntuar la sobrerrepresentación de un patrón o motivo lineal sobre el conjunto de proteínas que sirven como entrada del algoritmo desarrollado, asignando una “cantidad de información” contenida en cada motivo, calculada en función de la certidumbre de encontrar un determinado monómero en una posición determinada del motivo. Buscamos aquel motivo que maximice la cantidad de información sobre el conjunto de secuencias de entrada.
La combinación de un algoritmo goloso y de la teoría de la información molecular permitió en el pasado identificar y describir de forma cuantitativa el motivo lineal de ADN cuya presencia hace posible una interacción entre una proteína y el ADN genómico.
Computacionalmente, el ADN se representa con un alfabeto de cuatro caracteres, que son un subconjunto de los veinte caracteres con los que representamos a las proteínas.
La hipótesis de trabajo es que las interacciones mediadas por motivos lineales de proteínas y las interacciones mediadas por motivos lineales de ADN son equivalentes desde el punto de

vista formal y que, por lo tanto, podemos adaptar el algoritmo mencionado al estudio de motivos lineales de proteínas.
La validación de los resultados es doble: por un lado se contrastan los resultados obtenidos contra una base de datos de motivos ya validados mediante otros métodos aceptados por la comunidad; por otro lado, se hace una comparación del output de nuestro algoritmo con los resultados de un algoritmo de “fuerza bruta” para este mismo problema y para pequeños ejemplos en los que es posible ejecutar este método (de resultados exactos pero impracticable en ejemplos interesantes).
Como caso de estudio, se ejecuta el método para un conjunto de proteínas sobre las que se sabe que cumplen una función biológica determinada, buscando un motivo lineal que defina este comportamiento.

Abstract

The goal of the work presented here is to implement an algorithm for the identification and quantitative description of the linear motif that mediates the interaction between two proteins.
The biological problem we tackle is the identificatin of linear motifs responsible for physical interactions between proteins. In the last years, high-throughput experimental methods have identified a large number of physical interactions between proteins. Such methods yield the identity of the interacting proteins, but no details about the molecular interactions that lead to complex formation.
Many protein-protein interactions depend strongly on one or several linear motifs. Protein linear motifs consist of small fragments of a protein sequence, typically between 5 and 15 monomers long.
Published work by the groups of Thomas Schneider and Gary Stormo show that Shannon's information theory can be adapted for the study of molecular interactions between proteins and DNA linear motifs. The result of this adaptation is called "Molecular Information Theory".
In particular, the combination of a greedy algorithm and molecular information theory allowed the identification and quantitative description of the DNA linear motif that mediates the interaction between a protein and genomic DNA.
Our working hypothesis is that interactions mediated by protein linear motifs and interactions mediated by DNA linear motifs are equivalent from a formal viewpoint and, therefore, we can adapt the abovementioned algorithm to the study of protein linear motifs.
The algorithm receives an input the sequences for a set of proteins that interact with the same protein partner. The algorithm will search for overrepresented motifs within those sequences. For each putative motif, we will analyze the alignment composed of all examples of the motif present in the sequence set under examination. The amount of information contained in the alignment will help us estimate the degree of overrepresentation of the motif.
The results will be validated in two ways. On one hand, we will apply our algorithm on a database of motifs that have been characterized by well-accepted methods. On the other hand, we will compare the output of our algorithm with the results of a "brute force" algorithm applied to the same problem, for small datasets in which it can be applied (the "brute force" algorithm yields exact results but is impractical for examples of biological interest).
As a case study, we will apply the method to a set of proteins that localize in the nucleolus, searching for a linear motif responsible for this behavior.

Agradecimientos

Este trabajo marca el final de una etapa que abarcó varios años de mi vida. Una etapa fundamental, definitoria de quién soy. A lo largo de estos años diferentes personas me ayudaron, de una manera u otra, a estar donde estoy ahora y me gustaría agradecerles.
A mis viejos y mi hermana les agradezco el apoyo, y mucho más les agradezco el entendimiento sin condiciones ni cuestionamientos que siempre han tenido con mi complicada forma de ser.
A mi novia Laura, por apoyarme, entenderme y quererme, por compartir conmigo parte de este camino.
A Nacho por aceptar la dirección de esta tesis, la cual implicó aprender muchas cosas que siempre estuvo dispuesto a enseñarme. Me llevo mucho más que lo que se puede leer en este trabajo.
A Esteban y Cristina por aceptar ser jurados de este trabajo.
A toda la gente de LFP por los consejos para esta presentación.
A Martín, Sergio, Palla y Tara, por las vueltas en auto a la noche, por tantos y tantos tps hechos juntos, por las trasnoches estudiando, sin todo esto dudo que hubiese llegado este momento.
A Fran, Alan, Sole, Dai, Flor, Flora, Lucho, Darío, Nico, Mati, Lore, Diego por acompañarme, algunos desde siempre.
A Mati, Fer, Luigi, Facu, Bruno, Giga, Maxi, Murray, Leo R, Nati, Vivi, Javi, Nelson, Leo S, Zaiden, Eddy y no sé si me olvido de alguien, porque el grupo que se armó en estos años convirtió a la facultad en un lugar de amigos, además de un lugar de estudio.
A Luciano Moffatt y Verónica Becher, importantes en mi vida académica.
A esta universidad y en particular a esta facultad que me dieron una educación de calidad y la posibilidad de conocer el mundo académico.

Contenidos

Introducción al problema biológico

Las proteínas

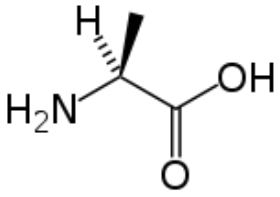
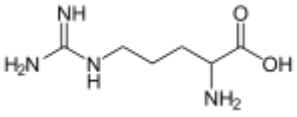
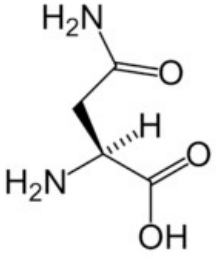
El presente trabajo estudia las interacciones entre proteínas. El presente capitulo será una introducción de los temas fundamentales de biología necesarios para la comprensión de este trabajo.

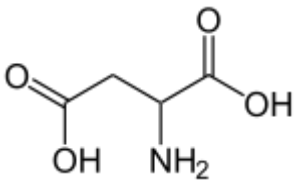
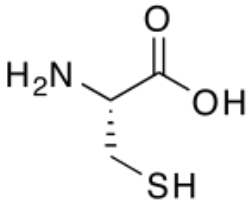
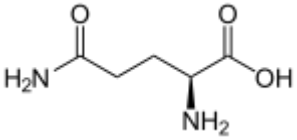
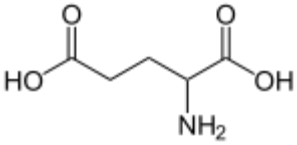
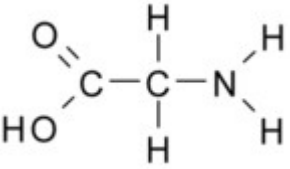
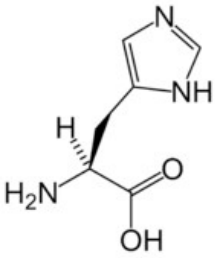
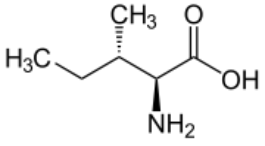
Existen compuestos biológicos de gran peso molecular, llamados macromoléculas, como por ejemplo las proteínas, los glúcidos y los acidos nucléicos. Estas macromoléculas son polímeros: están formadas por la unión molecular de una gran cantidad de moléculas más pequeñas que denominamos monómeros.

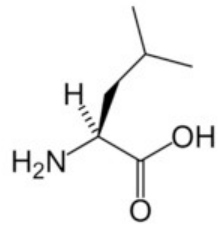
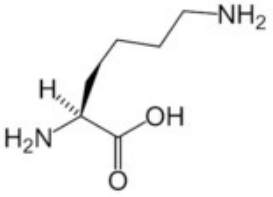
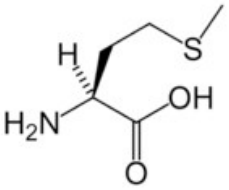
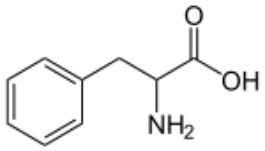
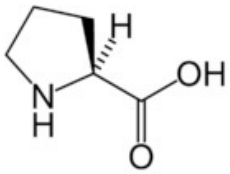
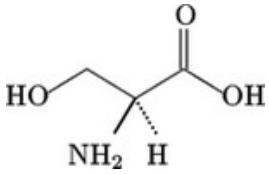
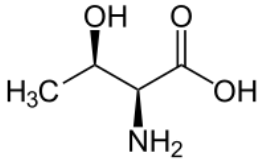
Cada tipo de macromolécula está compuesta por una clase distinta de monómeros. En el caso de las proteínas, los monómeros que las forman son veinte aminoácidos distintos [Bailey 1951]. Por lo tanto, la unión molecular de varios aminoácidos compone una proteína.

Para los aminoácidos existen dos códigos de representación: uno de tres letras y otro de una letra, que usaremos a lo largo de este trabajo.

Los aminoácidos poseen estructuras químicas comunes, que los definen como tales. También se observan partes de la estructura química que son diferentes, lo que permite distinguir a un aminoácido de otro.

Aminoácido	Código de 3 letras	Código de 1 letra	Estructura química
Alanina	Ala	A	
Arginina	Arg	R	
Asparagina	Asn	N	

Ácido aspártico	Asp	D	
Cisteína	Cys	C	
Glutamina	Gln	Q	
Ácido glutámico	Glu	E	
Glicina	Gly	G	
Histidina	His	H	
Isoleucina	Ile	I	

Leucina	Leu	L	
Lisina	Lys	K	
Metionina	Met	M	
Fenilalanina	Phe	F	
Prolina	Pro	P	
Serina	Ser	S	
Treonina	Thr	T	

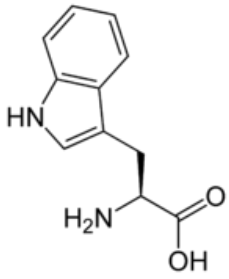
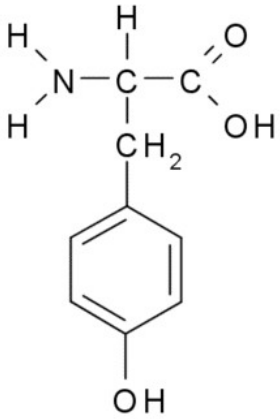
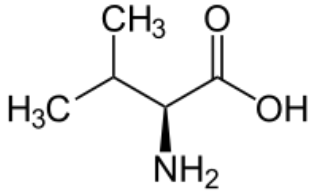
Triptófano	Trp	W	
Tirosina	Tyr	Y	
Valina	Val	V	

Tabla 1: Código y composición química de los aminoácidos.

La sucesión de aminoácidos en una proteína se denomina secuencia o estructura primaria [Oppel 1960]. En este trabajo analizaremos secuencias proteicas, que trataremos como una cadena de caracteres que en cada posición poseen alguna letra del alfabeto de 20 aminoácidos que pueden componer una proteína.

Las proteínas son encargadas de cumplir diversas funciones en los seres vivos, como mantener su estructura, regular procesos, recibir señales de factores externos o mediar la respuesta inmune. Las proteínas se organizan con una cierta estructura tridimensional, que hace posible su función [Perutz 1960]. Dicha estructura puede variar al modificar factores como la temperatura o el pH, lo que puede acarrear una modificación de las funciones de la proteína.
Dentro de la estructura de una proteína, pueden existir regiones que se pliegan sobre si mismas, formando dominios plegados con estructuras aproximadamente globulares, lo que vuelve poco accesible físicamente a esa porción de la secuencia. Se dice que estos dominios globulares poseen estructura terciaria. Por otro lado, existen porciones de la secuencia de la proteína que no se pliegan sobre sí mismas y permanecen accesibles a otros agentes [Uversky 2010]. Las llamamos regiones desplegadas o dominios desplegados.

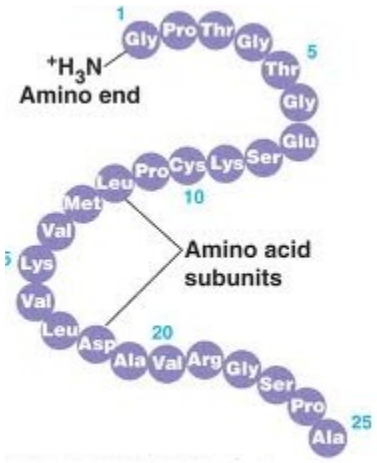


Figura 1: Estructura primaria de una proteína, donde se describe la secuencia de aminoácidos que la compone.

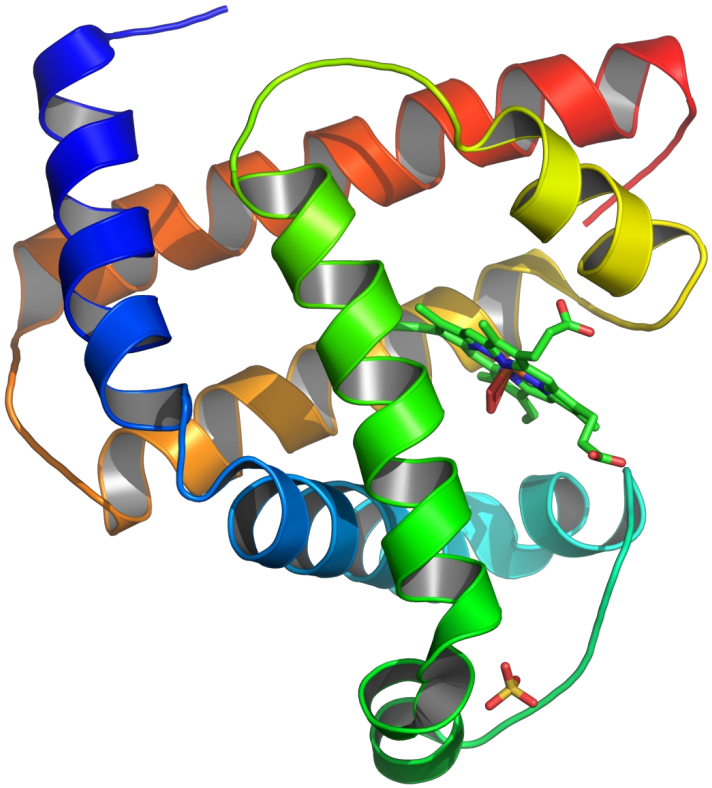


Figura 2: Representación de la estructura tridimensional de una proteína globular, en donde se puede observar su estructura terciaria compacta.

Interacciones entre proteínas

Las proteínas tienen la capacidad de interactuar entre si, lo que juega un papel importante en la función de muchas de ellas. Por ejemplo, la llegada de un estímulo externo a una célula modifica la red de interacciones entre proteínas, haciendo posible una respuesta al estímulo [Han 2004].
Las interacciones entre proteínas pueden presentarse entre un dominio plegado y un segmento de un dominio desplegado, o entre dos dominios plegados.

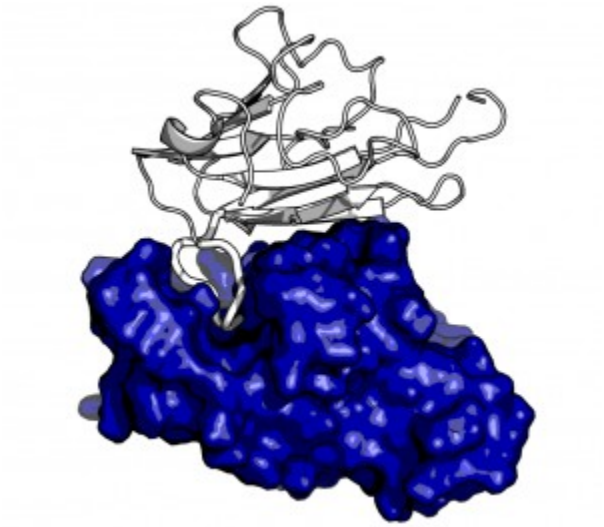


Figura 3: Interaccion entre dos dominios plegados de proteínas, representados como una superficie azul y un ovillo blanco para resaltar la complementariedad de sus formas.

En el presente trabajo estudiaremos las interacciones que se producen entre un dominio plegado y un segmento de un dominio desplegado. Este tipo de interacciones tienen importancia en eventos intracelulares activados por señales externas [Scott 2009]. Podemos describir el mecanismo de estas interacciones como el reconocimiento por parte del dominio plegado de señales dentro del dominio desplegado. Las señales reconocidas no son otra cosa que porciones de la secuencia de la proteína, que deben estar físicamente accesibles para que dicho reconocimiento tenga lugar. Tales porciones de secuencia se suelen denominar “sitios de unión” y “motivos lineales” [Gould 2009].
Cuando un sitio de unión forma un complejo con un dominio globular de otra proteína suele adoptar una conformación lineal. Así, los monómeros del motivo se alojan físicamente en cavidades separadas de la superficie de la proteína globular, también llamadas bolsillos [ver figuras 3 y 4]. Llamamos “motivo lineal” a la cadena de caracteres con la que expresamos cómo debe estar compuesto un sitio de unión, con letras representando monómeros y usando la letra x cuando cualquier monómero puede estar presente en la posición indicada [Aasland 2002].

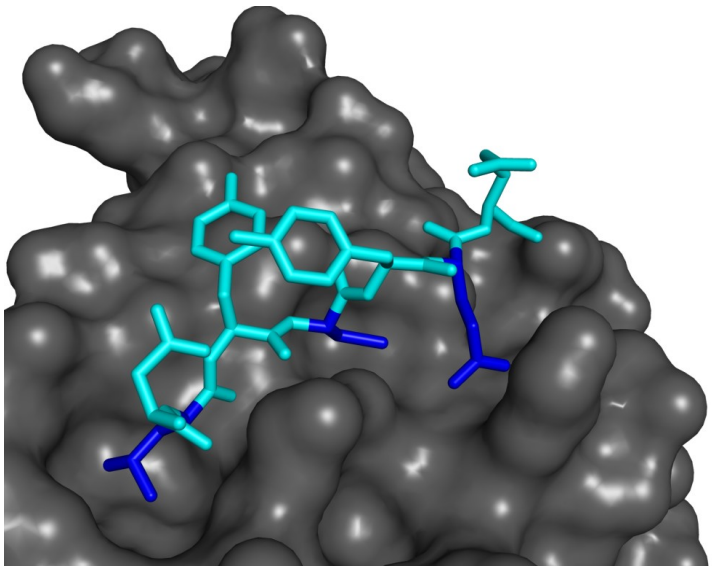


Figura 4: Modo unión entre la secuencia DLYCYEQ (en celeste y azul) y la proteína globular retinoblastoma (superficie gris). Los residuos L, C y E (en azul) son los determinantes estructurales de la interacción. El motivo lineal en este ejemplo sería LxCxE, donde las x pueden representar cualquier monómero.

Los motivos lineales de proteínas son abundantes y diversos: un reciente relevamiento incluye 1300 ejemplos caracterizados de 146 motivos lineales distintos [Gould 2009]. Según estimaciones [Neduva 2005] que toman en cuenta la cantidad de motivos funcionales hallados y la cantidad de interacciones que podrían estar mediadas por un motivo lineal puede decirse que aún quedan miles de motivos por explorar.
Las proteínas homólogas (que comparten un mismo origen evolutivo) tienen la propiedad de mantener con alto grado de probabilidad las funciones que realizan. Esto genera que la probabilidad de encontrar un mismo motivo funcional en secuencias homólogas sea alta [Dinkel 2007]. Llamamos conservación a este fenómeno.

Problema biológico

El problema biológico que abordaremos en el presente trabajo es el de las interacciones que se producen entre proteínas. Para muchas interacciones, carecemos de una descripción de los fundamentos moleculares de la interacción [Aloy 2005].
En el caso de algunos dominos globulares, se sospecha que interactúan mediante el reconocimiento de un “motivo lineal” porque se unen a dominios desplegados de otras proteínas.
En esta tesis tomaremos como punto de partida la secuencia de las proteínas que interactúan con un dominio globular dado y buscaremos cuál es el motivo lineal presente en las secuencias conformadas por dichas proteínas. La hipótesis es que dicho motivo es el responsable de la interacción [Neduva 2005].

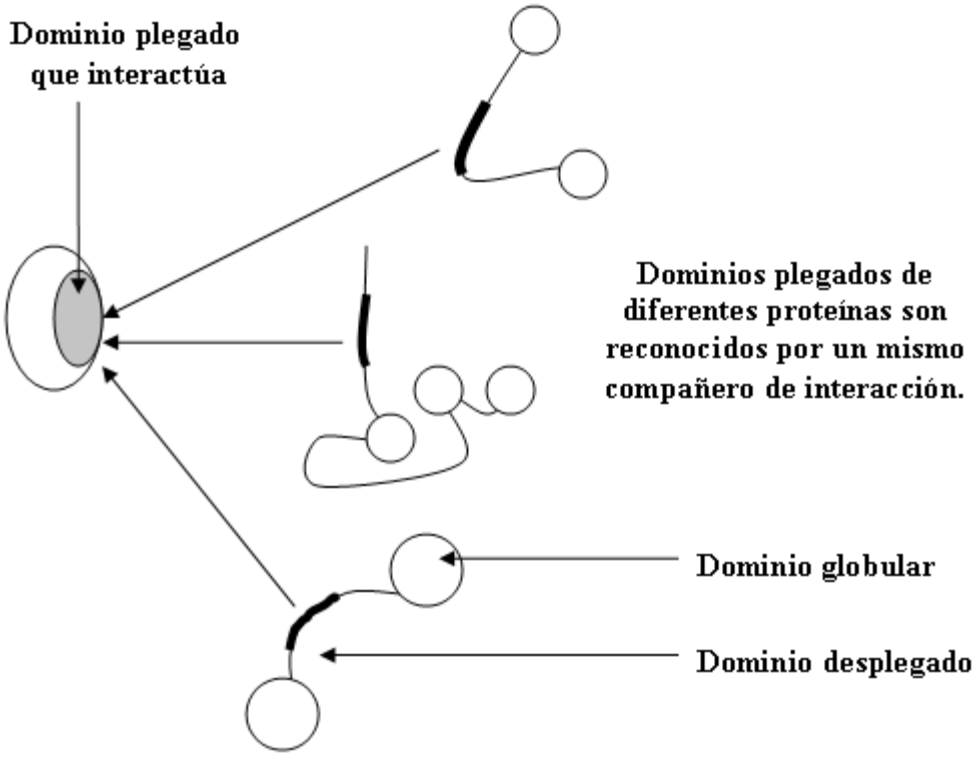


Figura 5: Interacción entre dominios plegados y desplegados.

De la biología a la computación

Representación de proteínas mediante secuencias

Para la resolución computacional del problema biológico planteado, necesitaremos plantear la forma en que los datos biológicos que poseemos serán representados computacionalmente.
Poseemos únicamente como datos de entrada, sobre los que trabajaremos en nuestro desarrollo, información de las secuencias de aquellas proteínas en las cuales estamos buscando un motivo común, el cual permite la interacción de ese conjunto de proteínas que está siendo analizado con una misma proteína o "compañero de interacción".
La manera en la que representaremos estos datos computacionalmente es mediante secuencias de caracteres que representarán las secuencias de aminoácidos que componen dichas proteínas. Utilizaremos para representar mediante caracteres a las proteínas el código de una letra especificado en la introducción de este trabajo.
Tenemos entonces un alfabeto para representar los datos compuestos por veinte posibles caracteres, los cuales se corresponden con los veinte aminoácidos que pueden componer la secuencia de una proteína.

Sobre estos conjuntos de secuencias de caracteres (nuestros datos de entrada) realizaremos operaciones como aplicar filtros para mejorar la calidad de los datos de entrada, aplicar el algoritmo desarrollado propiamente dicho, buscando el motivo anteriormente descripto.
También analizaremos los datos de salida mediante la composición de alineamientos, los cuales serán representados mediante conjuntos de "palabras" de igual longitud, compuestas de porciones de las secuencias de entrada, las cuales forman él o los motivos encontrados. Estos conjuntos de palabras serán utilizados para generar logos de secuencia [ver sección Representación mediante logos de secuencia] que permitan una interpretación de los datos.

Teoría de la información molecular

El algoritmo desarrollado en el presente trabajo, utiliza como herramienta que nos permite encontrar el motivo que permite la interacción entre proteínas, la Teoría de información molecular.
Estamos interesados en el hecho de calcular, para un alineamiento (conjunto de “palabras” de aminoácidos de igual longitud, en nuestro trabajo formadas por porciones de las secuencias de caracteres que representan las proteínas a ser analizadas) la cantidad de información que contienen.
Es mediante esta teoría que realizamos esté cálculo.

Definimos una máquina molecular como un sistema biológico capaz de ganar bits de información mediante la elección de dos o más estados posteriores partiendo de un mismo estado previo, ejecutando una función específica para un sistema viviente y disipando energía en el proceso [Schnieder 1997].
En el presente trabajo estamos interesados en mayor medida en el hecho de ganar información mediante la elección entre dos o más estados a partir de un mismo estado previo. Es decir, enfocamos el problema desde un punto de vista informático. Sin embargo, es interesante hacer notar que la disipación de energía en el proceso molecular tiene implicaciones que permiten relacionar la Teoría de la Información Molecular con la termodinámica y la teoría de la información clásica [Schneider 1991, 2010].

Cálculo de la cantidad de información contenida en un alineamiento de secuencias

Para dar una descripción completa de los alcances de la Teoría de la Información Molecular deberíamos introducir una gran cantidad de conceptos que no son los que este trabajo persigue.
A fin de no distraer al lector, nos centramos en el uso que hacemos de la Teoría de la Información Molecular.
Nuestro objetivo es encontrar motivos lineales sobrerrepresentados en un conjunto de secuencias. Mediremos esa sobrerrepresentación mediante el contenido en información del conjunto de secuencias que corresponden a cada motivo [Schneider 1990].

Nuestro objeto de estudio es una proteína que reconoce, o no reconoce, un mensaje presente en otra proteína. Estos mensajes están determinados por una determinada porción de secuencia, la cual es o no reconocida. Dado el conjunto de secuencias reconocidas por nuestra máquina, podemos alinear dichas secuencias y analizar el alineamiento en términos de las frecuencias con las que se presenta cada monómero posición a posición [Schneider 1990].
Como ejemplo, tomaremos el alineamiento del conjunto de secuencias {KRAT, GRAT, KRAR}. Podemos entonces medir la cantidad de información contenida en el alineamiento. En primera instancia calculamos la medida descripta por Shannon para valorar la incertidumbre:

$$H(l) = - \sum_{\forall\, m \in monomeros} f(m,l)\, log_2 f(m,l)\, [BIT] \qquad [1]$$

Donde H es la incertidumbre en la posición l del alineamiento, m es alguno de los 20 monómeros que puede formar parte de la secuencia de una proteína y f es la frecuencia de dicho monómero en la posición l.
En el caso de nuestro ejemplo tenemos entonces que sumar aquellos monómeros cuya frecuencia sea distinta de cero, por lo que obtendremos:

	Pos 1	**Pos 2**	**Pos 3**	**Pos 4**
Secuencia 1	K	R	A	T
Secuencia 2	G	R	A	T
Secuencia 3	K	R	A	R
Frecuencias	f(K) = ⅔ f(G) = ⅓	f(R)=1	f(A)=1	f(T) = ⅔ f(R) = ⅓
Incertidumbre (bits)	0.91	0	0	0.91
Información (bits)	3.41	4.32	4.32	3.41

Tabla 3: Análisis del calculo de información en secuencias de ejemplo

$$H(1) = -\,(2/3) * log_2(2/3) - (1/2) * log_2(1/3) \simeq 0.91\, bits$$

Los dos tercios corresponden, en la posición uno al monómero K, mientras que el tercio restante es la frecuencia del monómero G.

$$H(2) = -1 * log_2(1) = 0$$

En este caso solo tenemos un monómero cuya probabilidad es uno, el R. Vemos que la incertidumbre de la posición 3 será igual a la de la posición 2 y la de la posición 4 igual al de la

primera posición ya que las frecuencias son similares (solo cambian los monómeros sobre los cuales calculamos las frecuencias).

La información total de la posición es determinada por la fórmula:

$$R_{secuencia}(l) = X - H(l) \quad [2]$$

Donde *R* es la información de la posición, *l* es la posición del alineamiento sobre la que estoy calculando y *X* es el máximo valor de incertidumbre que puede alcanzar la posición dada (por ejemplo, en nuestro caso, al trabajar con proteínas, las cuales tienen 20 monómeros, *X* sería $\log_2(20) \approx 4.32$ bits).
En nuestro ejemplo, la información en las posiciones 2 y 3 es de 4.32 bits, mientras que en las posiciones 1 y 4 estará dada por el valor 4.32 - 0.91 ≈ 3.41 bits.
Con esto obtenemos el valor de información de cada una de las posiciones del alineamiento. Para obtener el valor de información total del alineamiento lo que debemos hacer es sumar el valor de información posición a posición. En nuestro ejemplo obtenemos valor aproximado total de 19.12 bits.

Factor de correción para el muestreo de n secuencias

El hecho de utilizar frecuencias en muestreos para calcular valores de información en lugar de probabilidades poblacionales lleva inevitablemente a una sobreestimación de la cantidad de información en los cálculos [Schneider 1986]. Dicha sobreestimación es grande cuando la cantidad de muestras con las que contamos es pequeña.
Para resolver este problema utilizamos *e(n)*, un factor de correción para el muestreo de n secuencias [Schneider 1986]. El valor de *e(n)* puede calcularse de una manera exacta, pero el cálculo solo es implementable de forma práctica en pequeños valores de n. También puede calcularse de manera aproximada. Es la forma que utilizaremos en este trabajo, útil cuando los valores de n crecen. Está dada por la fórmula:

$$e(n) = \left(\frac{s-1}{2\,ln\,(2)n}\right) \quad [3]$$

Donde s es la cantidad de símbolos de nuestro alfabeto (en nuestro caso los 20 monómeros posibles) y n es la cantidad de muestras con la que nos estamos manejando.
En el caso del ejemplo correspondiente al apartado anterior, estamos hablando de 3 muestras y 20 símbolos por lo que el factor de corrección es:

$$e(3) = \left(\frac{20-1}{2\,ln\,(2)3}\right) \simeq 4.56\,bits$$

Como podemos observar, cuando la cantidad de secuencias es demasiado pequeña, el factor de corrección anula el contenido de información cada posición en nuestra secuencia. A medida que la cantidad de muestras se incrementa, el factor de corrección disminuye.

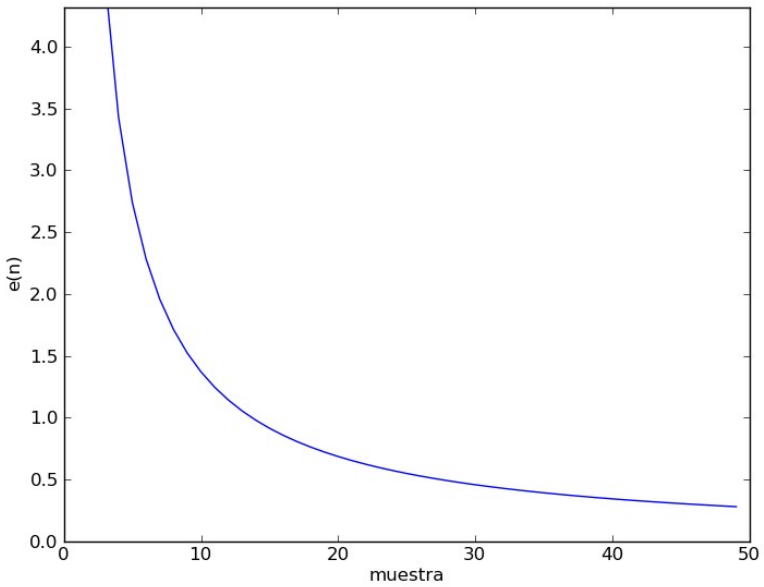


Figura 7: Valor de e(n) para un alfabeto de 20 símbolos en función de la cantidad de muestras.

Representación mediante logos de secuencia

Se puede mostrar la información contenida en un alineamiento de secuencias proteicas mediante el uso de logos de secuencia [Schneider 1990, Crooks 2004]. Este tipo de diagramas permiten una visualización más apropiada de cuál es el posible patrón que puede presentarse en un determinado alineamiento en comparación con una expresión regular del estilo de AxxA.
La altura de cada una de las posiciones es determinada por la información del alineamiento en dicha posición, utilizando las fórmulas descriptas en los apartados anteriores para calcular R(i). Por otro lado, la altura de cada una de los monómeros en una posición específica está determinada por la siguiente fórmula:

$$l = f(m,l)\, R_{sequencia}(l) \qquad [4]$$

En donde l es la altura que tendrá el monómero, f(m,l) es la frecuencia del monómero m en la posición l, y $R_{secuencia}(l)$ es la información contenida en la posición l de la secuencia. Aquel aminoácido con mayor probabilidad de aparición es colocado en el tope de la pila que cada posición representa en el gráfico, y luego se van colocando los demás aminoácidos siguiendo ese orden. Por ejemplo:

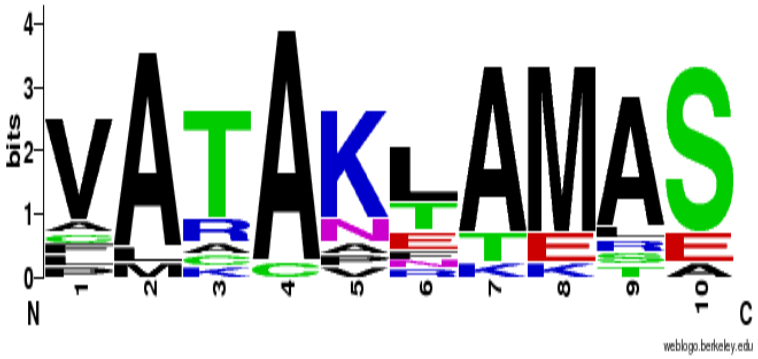


Figura 8: Ejemplo de logo de secuencia

Hemos utilizado la herramienta online [Crooks 2004] para generar el logo de secuencia correspondiente al ejemplo que venimos desarrollando a lo largo de esta sección obteniendo el siguiente gráfico como resultado:

Figura 9: Logo se secuencia generado con la herramienta WebLogo correspondiente al ejemplo desarrollado (alineamiento de las secuentas KRAT, GRAT y KRAR), puede observarse que las posiciones 2 y 3 tienen una altura de aproximadamente 4.32 bits, mientras que las posiciones 1 y 4 tienen una altura menor debido al aumento de la incertidumbre. Las letras tienen una altura correspondiente a su frecuencia de aparición en las posiciones correspondientes (lo que explica que, por ejemplo, el monómero K tenga el doble de altura que la letra G en la primera posición).

Validación de R(i)

La Teoría de la información molecular ha sido utilizada con éxito para cuantificar la conservación en secuencias de nuleótidos y proteínas [Schneider 1986]. Además, la teoría es útil para entender la relación entre estructura y secuencia en las interacciones ADN-Proteína [Schneider 2001].

De entre las magnitudes que se pueden calcular mediante la teoría de la información molecular, en este trabajo hacemos uso principalmente de R(i), el promedio de bits necesarios para definir un conjunto de secuencias alineadas. El uso de R(i) en un contexto biológico se ha validado de la forma siguiente [Schneider 1986]: se seleccionaron varias proteínas de unión a ADN bien estudiadas y se calculó el R(i) del alineamiento de sus sitios de unión en el ADN. R(i) mide la capacidad de estas proteínas, en tanto máquinas moleculares, para discriminar secuencias de ADN. Por otro lado, dada la longitud L del ADN del organismo en el que la proteína trabaja y el

número de sitios de unión N de la proteína, se calculó el número de bits R_{seq} ganado por una máquina que identifica correctamente N sitios en el contexto de un ADN de longitud L. R_{seq} representa la capacidad de discriminación que la proteína necesita para realizar su función. Se ha observado una correspondencia excelente entre R(i) y R_{seq} para diversas proteínas, lo que justifica el uso de R(i) para caracterizar conjuntos de secuencias alineadas.

Bases de datos de motivos funcionales

Buscaremos, como ya hemos indicado, para un conjunto de proteínas representadas mediante su secuencia de aminoácidos, los motivos funcionales que permiten la interacción de dicho conjunto con una determinada proteína sobre la que se conoce o presume que interactúan.
Existen en la actualidad bases de datos [http://elm.eu.org/] que contienen información de motivos de consenso para determinadas proteínas, mientras que otras contienen información sobre la secuencia de todas las proteínas conocidas [http://www.uniprot.org].
La información contenida en estas bases de datos nos servirá para hacer una validación de los resultados obtenidos por nuestro algoritmo [ver sección Validación del algoritmo sobre datos de relevancia biológica], las secuencias de las proteínas sobre las que se presume poseen un compañero de interacción común son obtenidos de la segunda base de datos, mientras que él o los motivos obtenidos por nuestro algoritmo son comparados con los motivos de consenso contenidos dentro de la primer base de datos.

Métodos actuales para encarar el problema

Existen diferentes maneras de encarar el problema biológico que hemos presentado, tanto experimentales como computacionales. Nosotros nos enfocaremos en aquellas que trabajan con las secuencias de las proteínas interactuantes para buscar los motivos lineales sobrerrepresentados en dichas secuencias.

Dilimot

La hipótesis central de este método es la misma que la de esta tesis: un conjunto de proteínas que posee un mismo “compañero de interacción” compartirán una característica que hará dicha interacción posible [Neduva 2005]. Dicha característica sería un motivo lineal presente dentro del conjunto de secuencias, detectable en virtud de su sobrerrepresentación.

Filtros sobre el Input

El conjunto de proteínas que comparten la interacción sirve como entrada del algoritmo. Sobre esta entrada, se aplican filtros. Solo el 15% de los motivos lineales conocidos ocurre en dominios globulares y segmentos transmembrana. Por lo tanto, se eliminan estas zonas.
Por otro lado se eliminan motivos lineales conocidos, fáciles de identificar y que no median interacciones entre proteínas, como las regiones de colágeno y los péptidos de señalización. Se trata de evitar su redescubrimiento, de poco interés cuando lo que se busca es un motivo responsable de interacciones entre proteínas.

También se comparan las secuencias para descartar la sobrerrepresentación de secuencias homólogas. Esto evita la sobrerrepresentación de motivos lineales conservados en secuencias homólogas.

El algoritmo

Al aplicar el algoritmo se buscan todos los motivos de largo entre 3 y 8. Se puntúa su sobrerrepresentación mediante la probabilidad binomial de encontrarla de manera aleatoria en un conjunto de secuencias similar [Neduva 2005, 2006]. Dicha probabilidad se computa de la siguiente manera:

$$P(n|M) = \binom{n}{M} p^{n}(1-p)^{M-n} \quad [5]$$

En donde p es la probabilidad de observar el motivo en la base de datos con la que se está trabajando, n es cuánto se ha observado el motivo en el conjunto de secuencias seleccionadas y M es el tamaño del conjunto.

Filtro sobre el output

Dilimot usa la hipótesis de que los motivos lineales funcionales son los que definen la función o una de las funciones que la proteína cumple y, por lo tanto, tienen mayor probabilidad de conservarse en proteínas homólogas, ya que estas suelen cumplir funciones similares o iguales. Por lo tanto, es esperable que aquellos motivos lineales sobrerrepresentados aparezcan en secuencias homólogas si se trata de motivos lineales funcionales.
El algoritmo busca proteínas homólogas a las del conjunto de partida y examina si los motivos lineales bajo consideración también están presentes en ellas. Se toman estas observaciones para computar una probabilidad. Finalmente, se multiplica esta probabilidad por la anterior para obtener una puntuación final que permite identificar los motivos lineales que están tanto sobrerrepresentados como conservados.

Resultados del algoritmo

Dilimot ha sido validado mediante la búsqueda de motivos lineales ya conocidos [Puntervoll 2003]. También se ha utilizado para descubrir nuevos motivos en conjuntos de secuencias no analizados anteriormente. De dicho análisis se obtuvieron motivos lineales candidatos a nuevos motivos funcionales, que fueron validados en estudios bioquímicos de reconocimiento entre proteínas.

SLimFinder

La sigla SLiM proviene de ”short linear motifs” (motivos lineales cortos). En la descripción del método [Edwards 2007], se especifica que el algoritmo se encuentra especialmente diseñado para identificar motivos compartidos por proteínas no relacionadas. Se menciona que en los métodos desarrollados previamente se realiza una búsqueda de motivos de forma genérica evaluando únicamente su sobrerrepresentación, lo que conlleva la necesidad de realizar una gran cantidad de procesamiento de manera posterior.

Filtros sobre el input

El algoritmo presenta varias opciones para filtrar los datos que le sirven como entrada.
Un primer filtro es la predicción de regiones desordenadas [Xue 2010], lo que permite acotar el tamaño de la entrada, eliminando regiones globulares sobre las que es improbable que se encuentren motivos funcionales.
Otro de los filtros que se aplica es la eliminación de regiones de baja complejidad, consistentes en la repetición de un mismo monómero una cantidad arbitraria de veces. En la descripción del algoritmo no se especifica que estas regiones carezcan de motivos lineales funcionales, solamente se aclara que son descartadas.
También son filtradas las apariciones del aminoácido Metionina al comienzo de cada secuencia. Las secuencias de proteínas comienzan por metionina y para los autores de este método esta aparición merece ser descartada.

El algoritmo

Se construyen los motivos de manera inicial mediante la combinación de “dímeros” generados a partir de una de las secuencias. Los dímeros consisten en dos aminoácidos separados por una o más wildcards. A continuación los dímeros son buscados en las demás secuencias a analizar. Son retenidos si tienen apariciones que justifiquen mantenerlos dentro de el conjunto de cantidatos. De otra manera son descartados.
Posteriormente se calculan las probabilidades de aparición de cada motivo, haciendo un ajuste que considera la cantidad total de motivos existentes. Una vez que se tienen los motivos puntuados por probabilidad se los devuelve como salida del algoritmo. Los resultados del algoritmo incluyen wildcards de tamaño variable.

Resultados del algoritmo

SLiMFinder ha sido testeado comparando sus resultados con los que arroja Dilimot sobre conjuntos de datos estudiados, obteniendo motivos similares a los arrojados por este. En muchos casos se obtienen motivos con menos wildcards, es decir, con mayor información acerca del motivo.

Método de Dinkel y Sticht

Este método no se centra en la búsqueda de motivos lineales nuevos, sino que analiza en profundidad la conservación de motivos lineales funcionales en secuencias homólogas. La idea básica se encuentra en identificar aquellos patrones que se conservan en mayor medida entre dos homólogos en comparación con lo esperado para cualquier porción de secuencia tomada al azar de un par de secuencias homólogas [Dinkel 2007].

Filtros sobre el Input

De manera análoga a lo que se mencionó en los métodos anteriores, se aplican métodos de enmascaramiento de regiones transmembrana y péptidos de señalización. También se extraen regiones globulares del espacio de búsqueda.

El Algoritmo

Los motivos son definidos en función de una base de datos preexistente, ELM [Gould 2009]. De allí se toman los motivos lineales funcionales cuya conservación se analiza.
Se calcula un puntaje de conservación promedio, que expresa la probabilidad de encontrar un motivo conservado entre un par de secuencias homólogas.

Filtro sobre el output

El puntaje de conservación promedio se calcula para conjuntos de secuencias homólogas progresivamente más divergentes. A medida que se agregan secuencias homólogas más divergentes, el puntaje de conservación promedio cambia su valor. El puntaje de conservación en un primer momento crece, hasta que las secuencias homólogas agregadas al estudio no aportan novedad y el valor comienza a decrecer. Posteriormente, el máximo puntaje de conservación (MPC) obtenido es usado para discriminar entre motivos funcionales y no funcionales. Los resultados obtenidos muestran que aquellos motivos que obtienen una mejor puntuación de conservación tienen más chances de ser motivos funcionales. Esta discriminación no se realiza de una manera taxativa: aquellos motivos que se se encuentran muy por encima del MPC se presume son funcionales y aquellos que están por debajo se presume que no lo son. En todo caso, el método aporta una valoración a cada motivo que luego debe ser analizada.

Resultados del algoritmo

Este método podría aplicarse sobre motivos lineales nuevos encontrados mediante métodos como Dilimot y SLiMFinder. La información obtenida serviría para distinguir aquellos motivos lineales que son funcionales.

Comparativa de los métodos nombrados

	Dilimot (2005)	SLiMFinder (2007)	D & S (2007)
Datos de entrada	Secuencias de proteínas	Secuencias de proteínas	Secuencia sobre la que ya existen datos sobre motivos y secuencias homólogas a ella
Objetivo	Descubrimiento de nuevos motivos	Descubrimiento de nuevos motivos	Análisis de la conservación de motivos
Filtros sobre los datos de entrada	Dominios globulares, segmentos transmembrana, regiones de colágeno, péptidos de señalización	Regiones desordenadas, regiones de baja complejidad, metionina como terminal	Dominios globulares, segmentos transmembrana, péptidos de señalización
Filtros sobre la salida	Análisis de conservación en secuencias homólogas	No	Análisis de conservación en secuencias homólogas
Wildcards en motivos	Longitud fija	Longitud variable	Longitud fija
Validación computacional y/o comparación con métodos anteriores	Validación contra motivos conocidos de la base de datos ELM	Validación contra motivos conocidos de la base de datos ELM y Dilimot	Validación contra motivos conocidos de la base de datos ELM
Validación experimental	Sí	No	No

Tabla 2: Comparativa de métodos analizados

Tras analizar los métodos existentes, hemos determinado qué tomar de lo que ya se ha hecho y qué cosas pueden ser mejoradas. Tomamos el uso de filtros, en el que se aprecia un consenso entre los trabajos anteriores. En nuestro caso, la aplicación de los filtros sobre las secuencias de entrada no se realizará de forma automática sino que se mantendrán como opciones para cada corrida individual [ver Sección Paramétros de Ejecución].

Encontramos que la representación de los motivos lineales es deficiente en los métodos aquí presentados [Schnieder 2002]. Un resultado posible de Dilimot o SLiMFinder es KxxDEL. Esta representación precisa cuál es la letra con mayor probabilidad de aparecer en cada posición o asigna una wildcard a la posición. Esto se contradice con los resultados experimentales, donde

con frecuencia más de una letra es apta para la función en cada posición, si bien unas son más eficientes que otras [Encinar 2009]. Esta observación sugiere la posibilidad de describir el motivo encontrado de una manera probabilística, que muestre la probabilidad de aparición de cada monómero en cada posición [ver Sección Logos de Secuencia].

Hipótesis de trabajo

La hipótesis del presente trabajo es que la Teoría de la Información Molecular puede ser utilizada para el estudio de las interacciones entre proteínas, ya que no existe una diferencia formal con las interacciones ADN-Proteína para la cual esta teoría ya ha sido utilizada con éxito.
Usaremos como datos las secuencias de los dominios desplegados de un conjunto de proteínas sobre las que se ha estudiado su interacción con un determinado dominio globular [Neduva 2005]. Aplicando una adaptación del algoritmo original descripto para las interacciones ADN-Proteína [Stormo 1989], obtendremos motivos lineales en función de su sobrerrepresentación. Estos motivos funcionarán como candidatos a ser los que permiten que la interacción se produzca.

Desarrollo algorítmico

Especificación del problema computacional

Antes de presentar la solución algoritmica propuesta para el problema de encontrar motivos funcionales que permitan la interacción entre proteínas, daremos una especificación de cuál es el problema computacional que estamos resolviendo.
Los datos de entrada para nuestro algoritmo, como ya hemos nombrado, están compuestos de un conjunto de secuencias de caracteres, los cuales representan la composición de las proteínas sobre las cuales el motivo será buscado. Estas secuencias de caracteres están definidas sobre un alfabeto de veinte letras, el cual se corresponde con el código de representación de aminoácidos con una letra, presentado en la introducción de este trabajo.
Los datos de salida de nuestro algoritmo serán alineamientos, definidos como conjuntos de palabras de un mismo largo (el largo del motivo buscado por nuestro algoritmo, definido como un parámetro para cada corrida individual del mismo) compuesto por porciones de secuencias de las proteínas de entrada.
El problema computacional que tratamos de resolver con nuestro algoritmo, es el de encontrar aquel alineamiento compuesto por palabras insertas en el conjunto de secuencias de proteínas del conjunto de entrada, que contiene la mayor cantidad de información (definida mediante la Teoría de la información molecular).
Este alineamiento podría calcularse de manera exacta, pero el costo computacional para un conjunto de entrada de tamaño razonable haría el cálculo impracticable. Para realizar una

validación de los datos de salida nuestro algoritmo, hemos desarrollado el método exacto y realizamos esa validación para un conjunto de entrada pequeño. Podemos ampliar entonces, y que el problema computacional que busca resolverse en este trabajo es encontrar el alineamiento que maximice la información en un tiempo razonable.
Como veremos más adelante, nuestro desarrollo se basa en una heurística que no garantiza resultados exactos. Por eso mismo, se realizan validaciones para comparar los resultados que arroja el algoritmo exacto con los que da nuestro algoritmo, y también validaciones para comparar los resultados de nuestro algoritmo con los resultados arrojados por otros que buscan resolver el mismo problema y con los motivos de consenso especificados en las bases de datos existentes.

El algoritmo original

El algoritmo original en el que se basa el presente trabajo fue definido por Gary Stormo en el año 1989 para encontrar motivos lineales de ADN que mediasen interacciones entre ADN y proteínas [Stormo 1989]. En el mismo se propone analizar secuencias de ADN para encontrar aquellos motivos sobrerrepresentados basándose en el contenido en información de los alineamientos candidatos. Básicamente, la propuesta es buscar el alineamiento de secuencias que contenga mayor cantidad de información.
El algoritmo es descripto de la siguiente manera:

- El usuario define k, el tamaño del motivo que estamos buscando.
- Cada uno de los substrings de largo k, o k-words, de la primera secuencia a ser analizada, constituirán un conjunto inicial de matrices interesantes. Para cada k-word se guardarán cuatro posiciones, representando cada posición a cada una de las cuatro bases que constituyen el alfabeto del ADN. Se indica con un 1 o un 0 si la base en la posición de su k-word correspondiente es ocupada por dicha base o no.
- La segunda secuencia es añadida al análisis, y cada integrante del conjunto de matrices interesantes es comparado con cada una de las posiciones de esta nueva secuencia. En cada una de las matrices interesantes se busca aquella posición de la nueva secuencia que maximice el contenido en información del alineamiento.
- Se genera un nuevo conjunto de matrices interesantes como resultado de haber agregado al análisis esta nueva secuencia. Este conjunto de matrices interesantes será usado para el paso siguiente.
- Se continúa con la siguiente secuencia hasta que no queden más secuencias por analizar.

Veamos un pequeño ejemplo concreto: queremos analizar cuál es el alineamiento de largo 3 que maximiza la información teniendo para analizar las cadenas de ADN “TATT y ”GATT”. Entonces, las k-words que generarán las matrices interesantes son “TAT” y “ATT”. Si la primera posición de la matriz la asignamos a las A, luego a las C, luego a las G y por último a las T, tendremos las siguientes matrices interesantes iniciales:

M_1= 0 0 0 1 = “T” M_2= 1 0 0 0 = “A”
1 0 0 0 = “A” 0 0 0 1 = “T”
0 0 0 1 = “T” 0 0 0 1 = “T”

Al agregar al análisis la segunda secuencia, debemos verificar cuál de las posiciones maximiza la información para cada una de las matrices interesantes. En el caso de la primera matriz, al intentar alinear “TAT” con “GAT” o con “ATT” obtenemos lo siguiente. En el alineamiento entre “TAT” y “GAT” los contenidos de información están dados la fórmula [2]:

$$R(1) = X - (H(1) + e(n))$$

Por el momento descartaremos el factor de correción e(n), ya que solo estamos interesados en saber cuál de los alineamientos maximiza la información. El factor de corrección es igual para ambos, por lo que se anula. El valor de X ya hemos dicho que para ADN es de 2 bits y el cálculo de H(1) utilizando [1] es el siguiente:

$$H(1) = -\sum_{\forall\, m \in monomeros} f(m,l)\, log_2 f(m,l) = (1/2 * log_2(1/2)) * 2 = 1$$

La sumatoria para todos los monómeros se anula en aquellos que la frecuencia es 0, mientras que existen dos monómeros para los cuales la frecuencia es ½. Por lo tanto, la información del primer sitio es de 1 bit. En los sitios 2 y 3 tengo que alinear letras idénticas, lo que hace que el factor de incertidumbre H se anule por tener frecuencia igual a 1, lo que nos da como resultado una información de 2 bits en cada posición. En total, sumando la información de cada una de las posiciones tendremos entonces 5 bits en el alineamiento.
El caso del alineamiento entre “TAT” y “ATT”, que es la otra posibilidad que debemos analizar, tendremos dos posiciones en los que la cantidad de información posicional es 1 bit. En la posición restante la información será de 2 bits, por lo que tendremos un total de información en el alineamiento de 4 bits.
De aquí se deduce que de las secuencias que analizamos, la que maximiza la información es aquella que incorpora “GAT” en el alineamiento, y por lo tanto generamos la nueva matriz interesante:

	Antes de la iteración	Después de la iteración
Alineamiento	TAT	TAT GAT
Matriz interesante	0 0 0 1 1 0 0 0 0 0 0 1	0 0 0.5 0.5 1 0 0 0 0 0 0 1
Información posicional	R(1) = 2 R(2) = 2 R(3) = 2	R(1) = 1 R(2) = 2 R(3) = 2

Tabla 5: Análisis del algoritmo original para un conjunto de secuencias de ejemplo

Esto es lo que ocurre para la primera de las matrices interesantes de nuestro algoritmo. Para la M_2 alineamos dos ocurrencias de “ATT”, por lo que no se ve modificada la matriz original y los valores se mantienen constantes.

Dentro de la matriz interesante guardamos las frecuencias de cada base en cada una de las posiciones. En cada posición del alineamiento de la matriz M_2 tenemos 2 bits, sumando un total de 6 bits, que superan los 5 bits totales de la matriz M_1’. Por lo que el motivo que el algoritmo devuelve como de información máxima es “ATT”.

Algoritmo desarrollado

Aspectos generales de la implementación

Para llevar a cabo la implementación de estre trabajo se utilizó el lenguaje de programación Python [www.python.org] teniendo en cuenta los siguientes aspectos:

- Al encontrarse el presente desarrollo en una etapa inicial, se buscó un lenguaje de implementación rápida que permitiese desarrollar en un tiempo razonablemente rápido. Esto ayudaría a su vez a analizar la viabilidad del método.
- Los conocimientos previos de Python de los involucrados en el proyecto.
- La popularidad del lenguaje en el ámbito de la biología y la existencia de bibliotecas desarrolladas para el lenguaje que resuelven múltiples problemas específicos a la bioinformática, como la lectura de archivos de secuencias biológicas o las librerías de gráficos de secuencias [Bassi 2007].

Input del algoritmo y aplicación de filtros

El desarrollo llevado a cabo implementa un algoritmo, que detallaremos más adelante. Adicionalmente, se utilizan algunas herramientas existentes para obtener una mejor calidad de datos en su entrada. Dichas herramientas son filtros sobre las secuencias de entrada, que permiten deshacerse de datos sobre los que potencialmente podemos no llegar a tener interés

(por ejemplo, regiones ordenadas). La aplicación de cada uno de los filtros es opcional y se especifica como parámetro en cada corrida individual del algoritmo.

Input

El input del algoritmo desarrollado está compuesto por un conjunto de secuencias de aminoácidos. Dichas secuencias corresponden a proteínas que comparten un determinado dominio globular como compañero de interacción. En la implementación de este proyecto hemos definido el input de esas secuencias para reconocer archivos FASTA [Pearson 1988].
Este formato de archivos es un standard de la disciplina y su organización es la siguiente:

- Por cada secuencia, incluiremos una línea cabecera que estará formada por el carácter '>' seguido del nombre de la secuencia.
- A continuación vendrá la cadena de aminoácidos de la secuencia actual. Este formato es utilizado también para secuencias de ADN.
- Se repite este esquema por cada secuencia del archivo de entrada.

El lenguaje Python posee una biblioteca con herramientas útiles para resolver problemas relacionados con la biología llamada BioPython [Cock 2009], la cual hemos utilizado para reconocer archivos con este formato y para guardar archivos intermedios resultado de la aplicación de filtros.

Agrupamiento de secuencias homólogas

Las secuencias homólogas, o de origen común, suelen presentar un alto grado de identidad de secuencia entre ellas [Rinderknecht 1976]. Este fenómeno no se limita a los motivos lineales funcionales. Por ello, no es recomendable incluir secuencias homólogas en el análisis ya que estaríamos sobrerrepresentando una gran centidad de secuencias conservadas que no son motivos lineales funcionales.
Hemos implementado un filtro que permite eliminar este tipo de redundancias presentes en el archivo de entrada. Usamos la herramienta CD-HIT [Weizhong 2006] para agrupar secuencias que superen un factor de similitud medido en percentiles y que es especificado como parámetro en cada corrida individual. Para cada grupo de secuencias, la secuencia más larga se proporciona como input al algoritmo y el resto se filtran.

Veamos un ejemplo:

Archivo de entrada	Agrupamiento 0.9	Agrupamiento 0.7	Agrupamiento 0.5
>Secuencia1 KAPALATRANSA >Secuencia2 KAPALATENSA >Secuencia3 KAPALATRETSA >Secuencia4 KAPALATRANSA	>Grupo1 KAPALATRANSA >Secuencia2 KAPALATENSA >Secuencia3 KAPALATRETSA	>Grupo1 KAPALATRANSA >Grupo2 KAPALATENSA	>Grupo1 KAPALATRANSA

Tabla 6: Resultados de la herramienta de agrupamiento de secuencias homólogas CD-HIT para un archivo. Se muestran los resultados de usar diferentes factores de agrupamiento.

Debemos señalar también que el uso de este filtro puede reducir notablemente el número de secuencias usadas como input, lo que aceleraría de forma significativa los pasos subsiguientes.

Predicción de regiones desordenadas

La mayor parte de los motivos lineales funcionales se encuentra en regiones desordenadas. Por ello, ofrecemos también la opción de aplicar el filtro de regiones desordenadas VSL2 [Vucetic 2003] sobre las secuencias de entrada. Las regiones desordenadas se proporcionan como input al algoritmo y las regiones ordenadas se filtran.

Como ejemplo, para la siguente secuencia:
MEPVDPRLEPWKHPGSQPKTACTNCYCKKCCFHCQVCFITKALGISYGRKKRRQRRRAHQNSQTHQASLSKQPTSQP
RGDPTGPKE

Obtenemos la siguiente predicción:

Predicted Disordered Regions:
1-19, 42-86
Prediction Scores:

NO.	RES.	PREDICTION	DISORDER	NO.	RES.	PREDICTION	DISORDER
1	M	0,835719	D	44	G	0,588375	D
2	E	0,839040	D	45	I	0,617112	D
3	P	0,830709	D	46	S	0,655010	D
4	V	0,810798	D	47	Y	0,685098	D
5	D	0,789160	D	48	G	0,710117	D
6	P	0,768471	D	49	R	0,733592	D
7	R	0,749609	D	50	K	0,755147	D
8	L	0,723353	D	51	K	0,775482	D
9	E	0,685243	D	52	R	0,791667	D
10	P	0,676291	D	53	R	0,806082	D
11	W	0,681410	D	54	Q	0,820135	D
12	K	0,671404	D	55	R	0,833594	D
13	H	0,664452	D	56	R	0,847465	D
14	P	0,672916	D	57	R	0,861596	D
15	G	0,676845	D	58	A	0,873783	D
16	S	0,653594	D	59	H	0,885695	D
17	Q	0,648853	D	60	Q	0,897136	D
18	P	0,621212	D	61	N	0,908479	D
19	K	0,560391	D	62	S	0,918276	D
20	T	0,497309	.	63	Q	0,926516	D
21	A	0,447150	.	64	T	0,932870	D
22	C	0,388213	.	65	H	0,942364	D
23	T	0,351939	.	66	Q	0,950123	D
24	N	0,330777	.	67	A	0,960393	D
25	C	0,326631	.	68	S	0,967914	D
26	Y	0,334602	.	69	L	0,974105	D
27	C	0,348943	.	70	S	0,980070	D
28	K	0,335784	.	71	K	0,984574	D
29	K	0,321746	.	72	Q	0,988847	D
30	C	0,319742	.	73	P	0,991266	D
31	C	0,305366	.	74	T	0,992831	D
32	F	0,324540	.	75	S	0,993701	D
33	H	0,328750	.	76	Q	0,994444	D
34	C	0,303035	.	77	P	0,994909	D
35	Q	0,343819	.	78	R	0,995607	D
36	V	0,377393	.	79	G	0,996219	D
37	C	0,361691	.	80	D	0,996509	D
38	F	0,371767	.	81	P	0,996784	D
39	I	0,390303	.	82	T	0,996917	D
40	T	0,438413	.	83	G	0,997096	D
41	K	0,463567	.	84	P	0,997096	D
42	A	0,533959	D	85	K	0,997058	D
43	L	0,557643	D	86	E	0,997024	D

Al ser las regiones desordenadas las que nos interesan, de toda la secuencia solo tendremos en cuenta las porciones 1-19 y 42-86 dentro de la totalidad de la secuencia. Dependiendo de las características del input, el tamaño de la entrada puede verse considerablemente reducido, dejando de lado porciones de secuencia donde probablemente no esté presente el o los motivos que estemos buscando.

Output del algoritmo y su tratamiento

El output que el algoritmo entrega es el conjunto de alineamientos que se encuentra utilizando su estrategia. Dicho conjunto nos llega en forma de matriz, la cual se encuentra, a su vez, ordenada en base a la cantidad de información que cada alineamiento tiene.
La forma en la que estos resultados son presentados al usuario es mediante archivos. Un archivo para cada alineamiento, numerado en el orden correspondiente a la cantidad de información. Dentro de cada archivo una tabla con los siguientes datos: posición, k-word y secuencia a la que pertenece.
Por ejemplo:

```
28     TTKKRTLRKNDRKKR     gi|5031869
42     TGKSKKKKKKKKQG     gi|223555917
44     KRRSKARKKRKKKSS     gi|115298645
7      LLRKQAWKQKWRKKG     gi|284005309
```

Estos datos resultan muy arduos para el análisis, por lo que además se utilizarán otras herramientas que detallaremos más adelante.

Correlación de alineamientos

En nuestro análisis se busca encontrar los alineamientos de k-words con la mayor cantidad de información. Cuando hablamos de alineamiento, estamos refiriéndonos a un conjunto de k-words de igual longitud pertenecientes a las secuencias de aminoácidos que forman parte del input del algoritmo. Los conjuntos que agrupen k-words similares serán los que presenten el mayor contenido en información.

Si dos alineamientos están compuestos por conjuntos similares de k-words, puede llegar a considerarse que estamos frente al mismo caso de análisis. Entonces, es conveniente crear a partir de ellas una nuevo alineamiento que contemple a todas las k-words contenidas en los alineamientos iniciales.

Hemos utilizado para la identificación de alineamientos similares el *coeficiente de correlación de Pearson* [Rodgers 1988]. Este coeficiente es útil a la hora de medir la relación lineal entre dos variables aleatorias. Tiene la ventaja, a diferencia de la covarianza, de ser independiente de la escala de medida de las variables. Una ventaja adicional es que toma un rango de valores acotado, entre -1 y 1, lo que justifica usar el mismo valor umbral para distintos casos de estudio.

El método utilizado es el siguiente:

- Generamos un vector para cada uno de los dos alineamientos que queremos comparar. El vector contendrá, para cada posición en el alineamiento, la cantidad de apariciones de cada uno de los aminoácidos.
- Calculamos el coeficiente de correlación de Pearson de los vectores generados.
- Si el valor obtenido supera el elegido por el usuario como umbral, tomamos los dos alineamientos y generamos uno nuevo con la unión de las secuencias de ambas alineamientos.

Como ejemplo, calcularemos el factor de correlación de los dos siguientes alineamientos:

Alineamiento 1	**Alineamiento 2**
AMYRT	AMYRT
MMYRT	MMYRA
AMERT	AMYET
AMYRA	
AEYRT	

Para el primer alineamiento, en la primera posición tendremos un 4 en la posición del aminoácido A, un 1 en el de M, y un cero en las 18 posiciones restantes. A continuación en el vector de cantidad de apariciones vendrán veinte posiciones más donde dieciocho serán ceros, teniendo un 4 la posicion de la M y un 1 en la posicion de la E. Realizamos la misma tarea para las tres posiciones restantes. De una manera similar armaremos el vector de apariciones en el alineamiento dos.

Los vectores resultantes para los alineamientos del ejemplo son:

Alineamiento 1: [4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0]
Alineamiento 2: [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0]

Un vez tenemos los vectores con la cantidad de apariciones de cada aminoácido en cada posicion, podemos calcular el coeficiente de correlación de Pearson. En el caso de nuestro programa, utilizamos NumPy [numpy.scipy.org], una biblioteca que ya tiene implementado el método para calcular este factor. El valor obtenido para nuestro ejemplo es de aproximadamente de 0.9342, lo que habla de una alta correlación entre ambos alineamientos.

Los dos alineamientos agrupados podrían tener k-words en común, que evitaremos incluir dos veces en el alineamiento final. Esto se debe a que en el caso de incluir una misma secuencia varias veces, a la hora del cálculo de información estaríamos ponderando en exceso esa k-word que aparece repetida. Cabe aclarar que nos referimos a una k-word tomada de una posición dada en una secuencia determinada, y no a dos k-words iguales en su secuencia de aminoácidos, pero tomadas de distintas posiciones en una secuencia o de dos secuencias distintas. En el ejemplo que acabamos de ver, al hacer la unión de los alineamientos deberíamos verificar si las dos apariciones de AMYRT que nos queda pertenecen o no a la misma secuencia en la misma posición. Si es así, eliminamos una de las apariciones; si no, dejamos ambas.

WebLogo

Descripción

WebLogo es una herramienta creada para la creación de “SequenceLogos”, o logos de secuencia [ver Sección Representación con Logos de Secuencia], de manera rápida y sencilla [Crooks 2004]. Se define como entrada de la herramienta un determinado alineamiento. La herramienta genera como salida un gráfico como el que ya hemos visto en la introducción del presente trabajo [Figuras 8 y 9].

Small Sample Correction

La utilización del factor de corrección, explicado en la introducción de este trabajo [ver Sección Factor de Corrección para muestreo de secuencias], es un parámetro de la herramienta de generación de gráficos. Dado que utilizamos este factor en los cálculos de nuestro algoritmo, lo empleamos también para la generación de los gráficos.

Errorbars

Otro parámetro que la herramienta posee es la posibilidad de mostrar en cada una de las posiciones del gráfico “errorbars”, o barras de error. Estas barras muestran el error estándar de la cantidad de información en cada una de las posiciones del alineamiento que estamos graficando [Schnieder 1997]. El error estándar de la muestra de n secuencias se calcula de la siguiente manera:

$$\sigma_{prom}(l) = \sigma_{R(l)} / \sqrt{n} \qquad [6]$$

La fórmula tiene en cuenta el desvío estándar de la muestra de n secuencias, sobre la raíz cuadrada de la cantidad de muestras.

Ejemplo de aplicación de WebLogo

Para el siguiente alineamiento,

```
{SLAMLAGARTAS,
SLAMLCGTRTAE,
SLAMLAGARTAS,
RLAKLAGARTAS,
SRAKLAGARTAS,
SLAMLAGARTAE,
SLAMLAGARTAS,
PLAMLCGTRTAE,
PLAMLAGARTAS,
RLAKLAGKRTAS,
SRAKLAGKRTAS,
SLAMLAGARTAE}
```

Este el el Sequence Logo asociado sin utilizar factor de corrección ni barras de error:

Figura 10: Sequence Logo sin factor de corrección ni errorbars.

Si agregamos el factor de corrección obtendremos el siguiente resultado:

Figura 11: Sequence logo con factor de corrección y sin errorbars.

En las posiciones 3, 5, 7, 9, 10 y 11 observamos una diferencia importante. En todas estas posiciones se observa un único aminoácido en nuestra muestra de doce secuencias. Al aplicar la corrección por el tamaño de la muestra, la altura de las letras disminuye del máximo (aproximadamente 4.32 bits) a unos 3.2 bits. Esto refleja nuestra incertidumbre acerca de la posibilidad de encontrar otros aminoácidos en dichas posiciones en muestras futuras.

Si agregamos a estos gráficos los errores estándar obtendremos la siguiente representación de la incertidumbre en la cantidad de información en cada posición del alineamiento:

Figura 12: Sequence Logo con factor
de corrección y errorbars.

Información en función de las rondas

Como salida adicional, se muestra un gráfico que grafica la evolución de la información con el pasar de las iteraciones del algoritmo para cada uno de los alineamientos. El gráfico muestra la curva correspondiente al alineamiento en cuestión, junto con las curvas de incremento de la información para todas los alineamientos que tienen mayor información que el actual.
En el siguiente ejemplo presentamos el gráfico para el sexto alineamiento con mayor cantidad de información (línea amarilla). Puede observarse también la información de otros cinco alineamientos que en el paso 73 cuentan con mayor cantidad de información (líneas azul, verde, fucsia, celeste y roja).

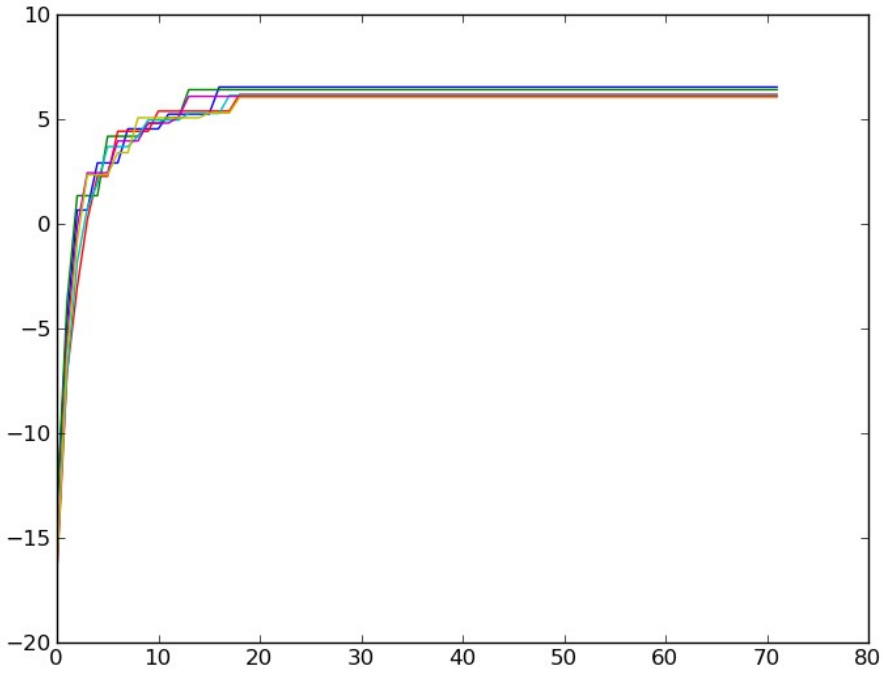


Figura 13: Evolución del la información de los alineamientos
en función de las rondas de corrida del algoritmo

Algoritmo

A continuación presentamos el pseudocódigo del algoritmo desarrollado.

```
input:
      Matriz M: conjunto de secuencias a analizar,
      Entero L: largo del patrón a buscar,
      Real P: factor de correlación utilizado para agrupar secuencias
similares

output:
      Matriz Res: Contiene todos los alineamientos de k-words(palabras
de largo L), ordenados por su cantidad de información.

algoritmo:
{
      M' = ObtenerTodasLasKWords(M)

      si (AlineamientosInicialesDeDosKWords?)
      {
            Res = CrearAlineamientosDeDosKWords(M')
      }
      si no
      {
            Res = CrearAlineamientosDeUnaKWord(M')
      }

      Mientras HayaCambiado(Res)
      {
            KWordsActuales = ObtenerTodasLasKWords(M)

            Para todo alineamiento A en Res
            {
                  AgregarKwordQueMaximiceInfo(A, KwordsActuales)
            }
      }
      si (AgruparPorCorrelacion?)
      {
            AgruparPorCorrelacion(Res, P)
      }

      OrdenarPorInfo(Res)

      Devolver Res
}
```

Descripción de las funciones auxiliares

Al considerar que es importante resaltar la funcionalidad del algoritmo desarrollado y no entrar en los detalles de la codificación, daremos una descripción de alto nivel de cada una de las funciones auxiliares que forman parte de la descripción del algoritmo.

- ObtenerTodasLasKWords: En el trabajo original de Stormo se toman las k-words de una secuencia y se la contrasta en busca de patrones con las k-words de la siguiente secuencia de entrada. Por ejemplo, si los datos de entrada están compuestos por las secuencias ACGMYAC y MAARTY y estamos buscando patrones de largo 5, el método compararía las k-words ACGMY, CGMYA y GMYAC contra las k-words MAART y AARTY. En el presente trabajo, se toman todos los datos de entrada como si fueran una sola gran muestra en la cual buscaremos patrones. Nosotros devolvemos comor resultado en esta función auxiliar las cinco k-words (ACGMY, CGMYA, GMYAC, MAART y AARTY) en un único conjunto y compararemos todas contra todas.
- CrearAlineamientosDeUnaKWord: Este método crea todos los vectores iniciales conteniendo en cada posición un vector con una única k-word. Esto se lleva a cabo para cada una de las k-words de entrada.
- CrearAlineamientosDeDosKWords: Este método crea todos los vectores iniciales conteniendo en cada posición un vector con dos k-words. Esto se lleva a cabo para cada par de las k-words del conjunto de entrada.
- AgregarKwordQueMaximiceInfo: Tomamos todos los alineamientos que se encuentren en el resultado hasta ese momento. Miramos una por una las k-words actuales de nuestro análisis y agregamos aquella cuya incorporación aumente más el contenido en información del alineamiento. Si no existe ninguna k-word que haga que la información aumente, entonces agregamos aquella en donde la información del alineamiento se vea menos perjudicada, pero solo permitiendo que la información descienda como máximo (5/n)%, en donde n es el número de ronda en el que se encuentra el algortimo. Si no existe k-word que cumpla con este requisito, entonces el alineamiento no se ve modificado. Adicionalmente, se controla que una misma k-word no figure dos veces en un mismo alineamiento y, dependiendo de las opciones de corrida, si existe solapamiento entre dos k-words de un alineamiento [ver Sección Parámetros de Corrida]. Se ha implementado un precálculo de información que permite acelerar el algoritmo desarrollado [Schneider 1996]. Consiste en almacenar en memoria el resultado de agregar cada posible aminoácido en cada una de las posiciones del alineamiento que esté analizando. Por ejemplo, si estamos buscando un alineamiento de largo cinco tendremos que calcular para cada uno de los alineamientos el resultado de agregar cada uno de los veinte aminoácidos en cada una de las cinco posiciones. En total habremos calculado y almacenado 100 resultados por alineamiento. Esto nos ahorra una gran cantidad de procesamiento en comparación con realizar el cálculo de información por cada k-word que se le ofrece al alineamiento, puesto que el número de k-words suele ser de al menos varios miles.
- AgruparPorCorrelacion: Toma todos los alineamientos de la matriz Res y agrupa aquellas cuyo factor de correlación sea mayor o igual a P en la forma explicada más arriba [ver Sección Correlación de alineamientos].

- OrdenarPorInfo: Ordena los alineamientos de la matriz Res en base a la cantidad de información.

Parámetros de corrida

Antes de comenzar cada corrida se deben especificar los siguientes parámetros:

- Nombre del archivo de datos que se quiere analizar.
- Largo del alineamiento que se busca descubrir.
- Cantidad de alineamientos de la matriz de resultados final que se quieren representar de forma gráfica (en orden descendente por cantidad de información).
- Aplicación de filtro CD-HIT, verdadero o falso.
- Identidad de secuencia utilizada para la aplicación del filtro CD-HIT. Toma valores entre 0 y 1, no se toma en cuenta si el parámetro anterior es falso.
- Aplicación de filtro VSL2, verdadero o falso.
- Aplicación de agrupamiento de resultados mediante análisis de correlación, verdadero o falso.
- Umbral para considerar dos alineamientos como suficientemente similares para agrupar mediante análisis de correlación. Toma valores entre 0 y 1, no se toma en cuenta si el parámetro anterior es falso.
- Aplicación del algoritmo exhaustivo, verdadero o falso.
- Permitir solapamiento de k-words, verdadero o falso. Si es falso, no se permite que dos k-words que se intersecan dentro de una misma secuencia formen parte del mismo alineamiento. El uso de este parámetro depende de las suposiciones previas acerca del mecanismo de acción de las proteínas involucradas.
- Permitir incluir dentro de un alineamiento múltiples k-words provenientes de una sola secuencia. El uso de este parámetro depende de las suposiciones previas acerca del mecanismo de acción de las proteínas involucradas.
- Armado de los alineamientos iniciales a los que luego potencialmente iremos agregando nuevas k-words. Puede realizarse a partir de todas las k-words disponibles del universo total de todos los posibles pares de k-words. Esta segunda opción vuelve más lento al método pero permite una exploración más profunda de los posibles alineamientos, ya que tenemos un comienzo exhaustivo previo a la aplicación de la heurística golosa.

Algoritmo exhaustivo

El algoritmo presentado es de naturaleza heurística, por lo que no garantiza encontrar los alineamientos de mayor información de entre todos los posibles. Para validar los resultados hemos desarrollado una versión exhaustiva del método que puede utilizarse para hallar resultados exactos. El algoritmo exhaustivo resulta poco eficiente y de uso impracticable para conjuntos de datos de un tamaño razonable, pero puede usarse sobre conjuntos de datos muy pequeños para realizar una comparación directa con el método heurístico. Presentamos a continuación el pseudocódigo.

```
input:
      Matriz M: conjunto de secuencias a analizar,
      Entero L: largo del patrón a buscar,
      Real P: factor de correlación utilizado para agrupar secuencias
similares

output:
      Matriz Res: Contiene todos los alineamientos de k-words,
ordenados por su cantidad de información.

algoritmo:
{
      M' = ObtenerTodasLasKWords(M)
      Res = CrearAlineamientos(M')

      Repetir #ObtenerTodasLasKWords(M) veces
      {
            KWordsActuales = ObtenerTodasLasKWords(M)

            Para todo alineamiento A en Res
            {
                  Res=AgregarNuevosAlineamientos(KWordsActuales,A,Res)
            }
      }

      OrdenarPorInfo(Res)

      Devolver Res
}
```

Vemos que la diferencia fundamental con el algoritmo presentado previamente es que aquí la estrategia no es golosa sino exhaustiva. Antes el número de alineamientos se mantenía constante. Se agregaba a cada alineamiento aquella k-word que aumentaba más la información, o ninguna si ninguna k-word aumentaba la información. En este caso, se agregan al análisis todos los nuevos alineamientos resultado de todas las posibles combinaciones de alineamientos actuales con las k-words a analizar. Esto no depende del contenido en información resultante.
Obviamente, la cantidad de alineamientos que forman parte de la matriz Res explota rápidamente en función del tamaño del conjunto de secuencias que se analiza. Es al ordenar en función de la cantidad de información que podremos identificar y representar en forma gráfica el alineamiento con mayor contenido en información.

Validación del algoritmo desarrollado

Comparación entre el algoritmo exhaustivo y el heurístico

Como primer método de validación del algoritmo desarrollado, de naturaleza heurística, lo comparamos con el algoritmo exhaustivo. El resultado es una primera valoración de la capacidad del algoritmo heurístico de encontrar los alineamientos de mayor contenido en información a partir de las secuencias de partida.
El conjunto de secuencias a analizar es el siguiente:

{KKKKKSLAM,
KKKKKCERN,
KKKKKTJYS,
KKKKKEJST,
KKKKKHSSC}

En este conjunto de secuencias buscaremos el motivo mejor puntuado de largo cinco. El resultado con mayor puntuación ha sido el siguiente en ambos métodos:

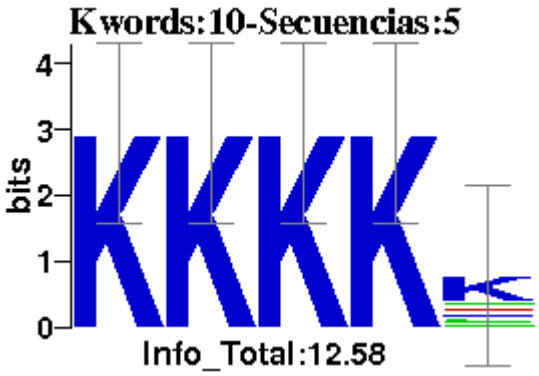


Figura 14: Motivo con mayor cantidad de información encontrado por ambos algoritmos para la secuencia de ejemplo

Conformado por el siguiente alineamiento:

KKKKC
KKKKS
KKKKT
KKKKE
KKKKH
KKKKK
KKKKK
KKKKK
KKKKK
KKKKK

Es decir, el alineamiento de largo cinco con mayor contenido en información contiene las cinco palabras de cinco lisinas (K) y las cinco palabras que poseen las cuatro primeras posiciones con lisinas y una quinta posición con otro aminoácido. El hecho de que los dos métodos obtengan el mismo resultado muestra que el método de búsqueda golosa implementado en el algoritmo heurístico funciona de manera similar, al menos en el conjunto de datos examinado.

Resulta ilustrativo señalar que en ambos algoritmos el alineamiento con mayor contenido en información aparece repetido en los resultados. La causa es que existen múltiples formas de construir el alineamiento, que se diferencian entre sí en el orden de incorporación de las k-words que lo forman. Esta clase de redundancia desaparece de los resultados cuando se agrupa alineamientos utilizando el análisis de correlación. Esto permite una mejor inspección visual de los resultados si uno está interesado no solo en el alineamiento de mayor contenido en información sino en examinar múltiples alineamientos con alto contenido en información.

Los resultados anteriores se obtuvieron permitiendo el solapamiento de k-words. Si no lo permitimos, tanto el método exhaustivo como el heurístico obtienen el siguiente resultado:

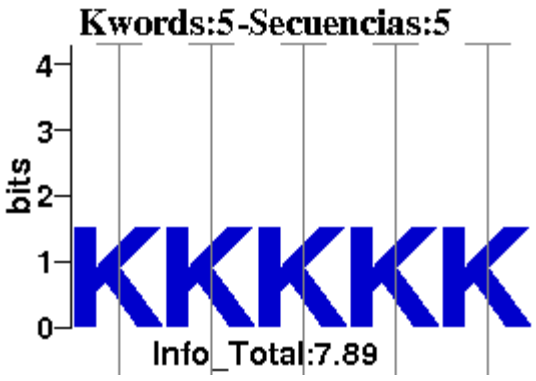


Figura 15: Motivo de mayor información encontrado en ambos algoritmos sin permitir el solapamiento de k-words.

Conformado por el siguiente alineamiento:

KKKKK
KKKKK
KKKKK
KKKKK
KKKKK

Nótese que el resultado obtenido está compuesto de las cinco palabras de cinco lisinas seguidas. Pese a que estas palabras son más similares entre ellas que las que conformacn el resultado mostrado más arriba, el contenido en información del alineamiendo se ve disminuido. Esto se debe a la baja cantidad de k-words que componen este alineamiento, lo que incrementa el factor de corrección. En todo caso, la coincidencia de resultados entre el algoritmo exhaustivo y el heurístico indica que este último funciona de manera simiar, al menos en el conjunto de datos examinado.

Validación del uso del algoritmo sobre datos de relevancia biológica

Si bien es importante que el algoritmo heurístico presentado reproduzca los resultados de una búsqueda exhaustiva, desde el punto de vista bioinformático son necesarias validaciones adicionales para refrendar la utilidad práctica del trabajo realizado. En primer lugar, debemos mostrar que nuestro algoritmo es capaz de encontrar motivos lineales funcionales conocidos dentro de un conjunto de secuencias. Este tipo de validación se ha realizado en el desarrollo de otros métodos para la búsqueda de motivos lineales en proteínas, como Dilimot y SLiMFinder (ver introducción). La tabla 8 muestra los resultados de dicha validación para el método Dilimot [Neduva 2005].

En nuestro caso, hemos acudido a la base de motivos lineales de proteínas ELM [Gould 2009]. Extrajimos motivos lineales conocidos con más de cuatro ejemplos catalogados. Para cada motivo, se extrajo de la base de datos Uniprot [Magrane 2011] la secuencia completa de la proteína. Las secuencias de todas las proteínas que contienen un motivo dado se usaron como datos de entrada del algoritmo. Tomamos motivos utilizados en la validación de Dilimot para hacer posible una comparación directa con los resultados de dicho método, el algoritmo de mayor aceptación actual dentro de la comunidad bioinformática. A continuación describiremos los resultados para 12 motivos lineales.

Los resultados que presentamos son provenientes de corridas donde aplicamos tanto CD-HIT (con umbral de 63% de similitud) como VSL2. No permitimos solapamiento de palabras dentro de los alineamientos. Corridas paralelas cambiando los valores a estos parámetros obtienen resultados similares, aunque de menor calidad. En esos casos, se observan aproximadamente los mismos motivos, pero en alineamientos con menor cantidad de información o relegados

debajo de otros motivos que ven incrementada su información por no aplicar filtros o permitir solapamiento.

Motivo “14-3-3 Tipo 1”

El motivo consignado en la base de datos es R(SFYW)xSxP. Dilimot encuentra el motivo RSxSxP en primera posición. El alineamiento de mayor contenido en información obtenido con nuestro método es el siguiente:

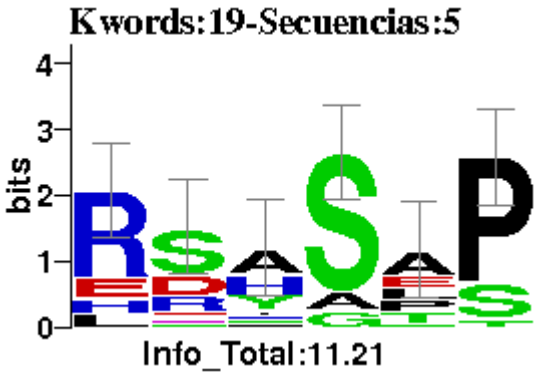


Figura 16: Motivo mejor puntuado hallado para “14-3-3 Tipo 1”

Observamos que tanto Dilimot como nuestro algoritmo obtienen resultados compatibles con el motivo conocido. Nuestro algoritmo sugiere que el motivo es significativamente más flexible que lo indicado tanto por Dilimot como por ELM. Por ejemplo, en la posición 4 se admite una alanina (A), además de una serina (S).

Motivo “gamma-adaptin”

El motivo consignado en la base de datos es (DE)(DES)xFx(DE)(LVIMFD). Dilimot encuentra el motivo DDxFxxF en primera posición. El alineamiento de mayor contenido en información obtenido con nuestro método es el siguiente:

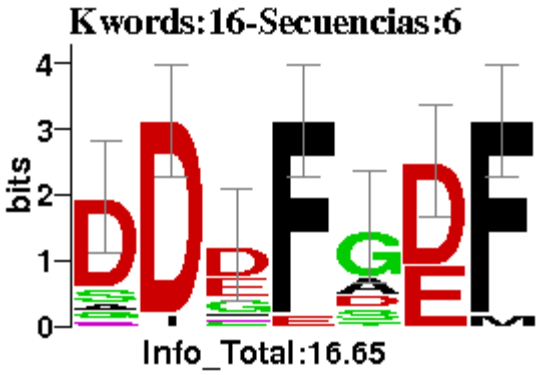


Figura 17: Motivo mejor puntuado hallado para “gamma-adaptin”

Observamos que tanto Dilimot como nuestro algoritmo obtienen resultados compatibles con el motivo conocido. Nuestro algoritmo capta mejor el motivo consignado por ELM en la posición seis.

Motivo “Clathrin box”

El motivo consignado en la base de datos es L(ILM)x(ILMF)(DE). Dilimot encuentra el motivo LIxLD en primera posición. El alineamiento de mayor contenido en información obtenido con nuestro método es el siguiente:

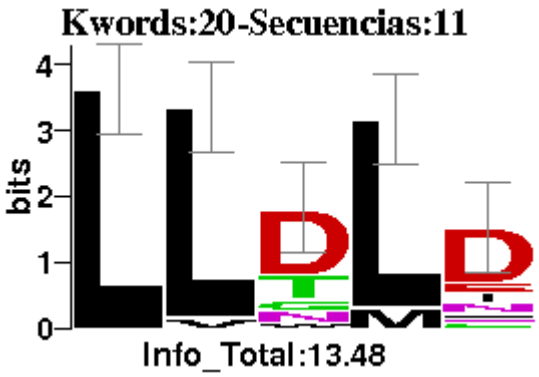


Figura 18: Motivo mejor puntuado hallado para “Clathrin Box”

Observamos que tanto Dilimot como nuestro algoritmo obtienen resultados compatibles con el motivo conocido. Nuestro algoritmo capta mejor el motivo consignado por ELM, si bien solamente en la posición cuatro.

Motivo “Integrin”

El motivo consignado en la base de datos es RGD. Dilimot encuentra el motivo RxDV. Nuestro método encuentra el siguiente alineamiento en la posición dos:

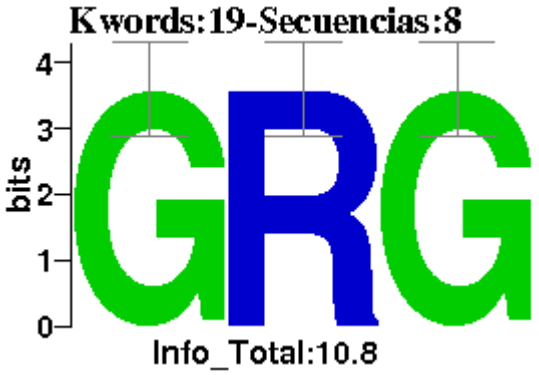


Figura 19: Motivo mejor puntuado hallado para “RGD”

Observamos que tanto Dilimot como nuestro algoritmo fallan a la hora de encontrar ese motivo en las secuencias de entrada. Dilimot encuentra un motivo que coincide con el reportado en la primera y tercera posición, mientras que nuestro algoritmo encuentra un motivo que coincide con el reportado en la primera y segunda posición.

Motivo “CtBP”

El motivo consignado en la base de datos es Px(DEN)L(VAST). Dilimot encuentra el motivo DxPxDL en primera posición. Nuestro método encuentra el siguiente alineamiento en la tercera posicion:

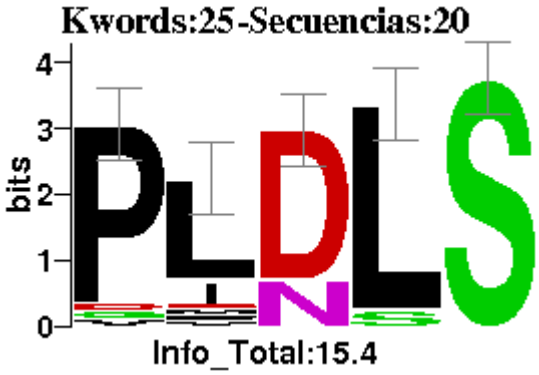


Figura 20: Motivo hallado
para “CtBP” en tercera posición.

El cual claramente se asimila más al motivo de consenso en comparación al arrojado por Dilimot.

Motivo “EH 1”

El motivo consignado en la base de datos es Fx(IV)xx(IL)(ILM). Dilimot encuentra el motivo FxIxNI en primera posición. Nuestro método no ha encontrado resultados satisfactorios en las primeras posiciones. A modo ilustrativo mostramos el resultado con mayor puntuación:

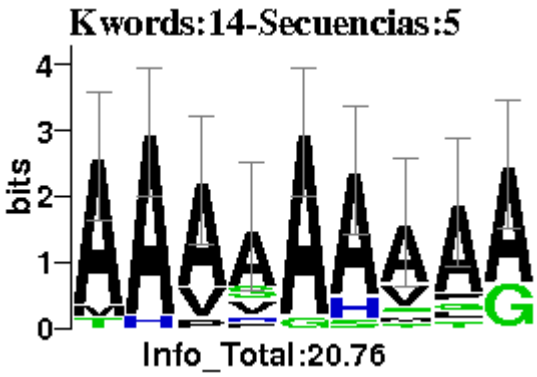


Figura 21: Motivo mejor puntuado hallado para “EH 1”

Vemos que no existe correspondencia alguna con el motivo buscado.

Motivo “HP-1”

El motivo consignado en la base de datos es PxVx(LM). Dilimot encuentra el motivo KVPxVxL en cuarta posición. Aquí tampoo hemos obtenido resultados favorables con nuestro método. Mostramos el alineamiento con mayor contenido de información:

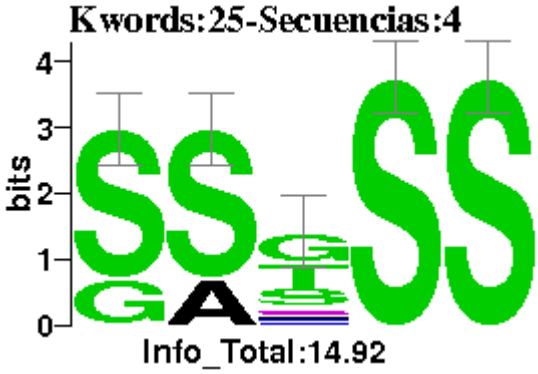


Figura 22: Motivo mejor puntuado hallado para “HP-1”

Motivo “Mannosylation”

El motivo consignado en la base de datos es WxxW. Dilimot encuentra el motivo DGxW en primera posición. Nuestro método ha obtenido en segunda posición el siguiente alineamiento:

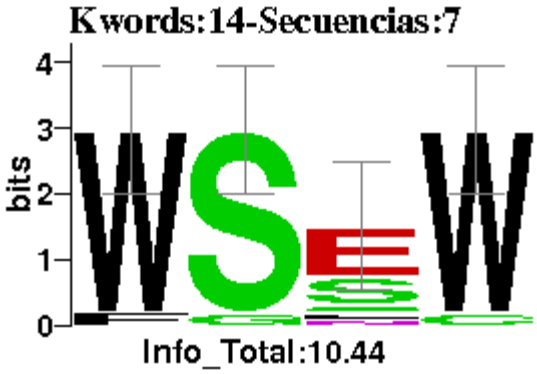


Figura 23: Motivo mejor puntuado hallado para “Mannosylation”

Lo que marca una certidumbre casi total en el primer y cuarto aminoácido, mientras que el segundo aminoácido está claramente más definido que en el consenso consignado en la base de datos.

Motivo “Dynein”

El motivo consignado en la base de datos es (KR)xTQT. Dilimot encuentra el motivo KxTQT en primera posición. Nuestro método ha obtenido en segunda posición el siguiente alineamiento:

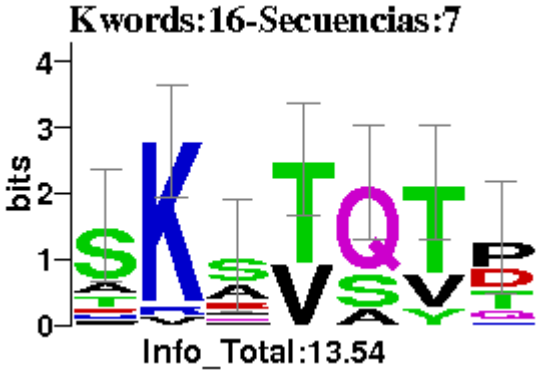


Figura 24: Motivo segundo mejor puntuado hallado para “Dynein”

El motivo hallado se corresponde con el motivo consignado en la base de datos.

Motivo “TRAF6”

El motivo consignado en la base de datos es PxE. Dilimot encuentra el motivo PQE en primera posición. Nuestro método ha obtenido en primera posición el siguiente alineamiento:

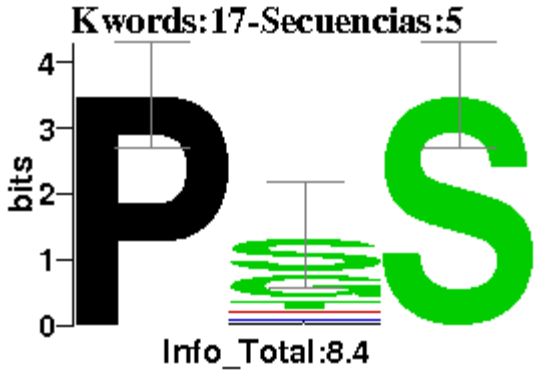


Figura 25: Motivo segundo mejor puntuado hallado para “TRAF6”

El motivo hallado es de peor calidad que el de Dilimot ya que se pierde la definición sobre el tercer aminoácido.

Motivo “NRBOX”

El motivo consignado en la base de datos es LxxLL. El método Dilimot no reporta resultados significativos para este motivo. Hemos ejecutado nuestro método y tampoco hemos obtenido resultados significativos.

Motivo “Retinoblastoma”

El motivo consignado en la base de datos es (LI)xCx(DE). Dilimot encuentra el motivo LxCxE en primera posición. No hemos obtenido resultados favorables con nuestro método. Mostramos el alineamiento con mayor contenido de información:

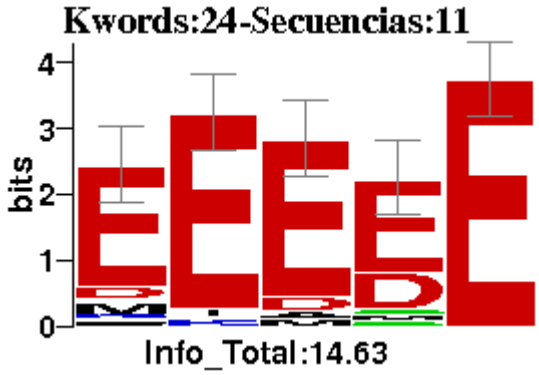


Figura 26: Motivo segundo mejor puntuado hallado para “Retinoblastoma”

Resulta interesante señalar que el motivo retinoblastoma se encuentra seguido en muchos casos de una serie de residuos de ácido glutámico (E) o ácido aspártico (D) [Chemes 2010], por lo que el resultado obtenido posee relevancia biológica.

Comparativa de resultados

Nombre del motivo	Motivo de consenso	Correspondencia entre Dilimot y el consenso (Ranking)	Correspondencia entre nuestro método y el consenso (Ranking)
14-3-3 Tipo 1	R(SFYW)xSxP	Buena (1)	Buena (1)
Gamma-adaptyn	(DE)(DES)xFx(DE)(LVIMFD)	Buena (1)	Buena (1)
Clathrin-box	L(ILM)x(ILMF)(DE)	Buena (1)	Buena (1)
Integrin	RGD	Parcial (2)	Parcial (2)
CtBP	Px(DEN)L(VAST)	Buena (1)	Buena (3)
EH 1	Fx(IV)xx(IL)(ILM)	Parcial (1)	No hallado
HP-1	PxVx(LM)	Buena (4)	No hallado
Mannosylation	WxxW	Parcial (1)	Buena (2)
Dynein	(KR)xTQT	Buena (1)	Buena (2)
TRAF6	PxE	Buena (1)	Parcial (1)
NRBOX	LxxLL	No hallado	No hallado
Retinoblastoma	(LI)xCx(DE)	Buena (1)	No hallado

Tabla 8: Motivos estudiados, sus respectivos motivos de consenso, el motivo encontrado por Dilimot en cada caso y el y correspondencia con los resultados de Dilimot y de nuestro algoritmo.

Podemos señalar que en los casos en los que el motivo no ha sido hallado, los motivos con altos puntajes de información se corresponden con regiones de baja complejidad. Como resultado de este análisis podríamos destacar que los motivos hallados tienen mayor cantidad de información que los motivos que se estaban buscando.
Por ejemplo, en el caso de “Retinoblastoma” el motivo de consenso es (LI)xCx(DE), lo que idealmente contiene una información de 4.32 en la posición 3 y de 4.32/2 en las posiciones restantes. Esto da como resultado aproximadamente 8.64 bits de información. Por otro lado, en el motivo hayado por Dilimot tenemos LxCxE, lo que serían idealmente tres posiciones de

aproximadamente 4.32 bits, lo que corresponde a 12.96 bits totales. Como podemos ver, el motivo hallado por nuestro método tiene 14.63 bits, un motivo de baja complejidad que supera en cantidad de información al motivo buscado. Basados en la literatura, presumimos que se trata de un motivo funcional que complementa al motivo buscado [Chemes 2010].

Caso de estudio

En muchos casos, nuestro algoritmo es capaz de recuperar al menos parcialmente, los motivos lineales conocidos a partir de las secuencias donde se encuentran. Su desempeño es de calidad similar a la de Dilimot, si bien es destacable que en algunos casos se obtiene una descripción más fina del motivo. Si bien será necesaria una validación más extensiva, nos pareció que la calidad de los resultados justificaba una primera aplicación del algoritmo a un problema biológico relevante. Esta aplicación se realizó en colaboración con la Dra. María Fátima Ladelfa y el Dr. Martín Monte, del laboratorio de Biología Molecular y Celular del Departamento de Química Biológica, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires.

Introducción al problema

Como introducción al problema que estudiaremos, creemos pertinente dar a conocer un esquema simplificado de la estructura de una célula eucariota.

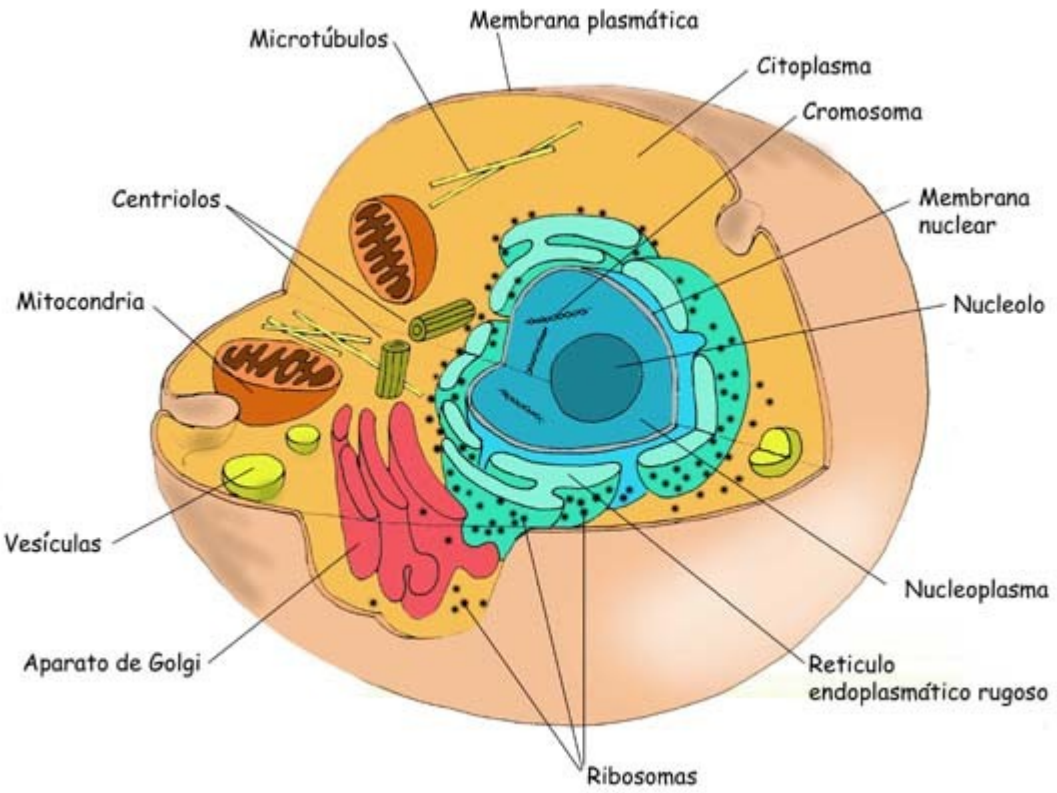


Figura 27: Estructura de una proteína eukariota.

La célula eucariota puede entenderse como un conjunto de compartimentos delimitados por barreras físicas. Entre ellos se encuentran el citoplasma y el núcleo. En el citoplasma de la célula se producen proteínas que cumplen una variedad de funciones y que, luego de ser producidas, se mueven a diferentes lugares dentro y fuera de la célula.
En este estudio, tendremos en cuenta un conjunto de proteínas para las cuales se ha estudiado empíricamente que ingresan al núcleo y allí se incorporan a la subestructura llamada nucleolo [Emmot 2009,Scott 2010,Borysko 1951]. Estas proteínas tienen en común su localización celular pero no la función que realizan.
No se conoce bien el mecanismo de incorporación de proteínas al nucleolo [Emmot 2009,Scott 2010]. Sí se conoce que la incorporación de otras proteínas a ciertas estructuras celulares depende en ocasiones de motivos lineales [Rastogi 2010]. Por ello se ha realizado un estudio utilizando nuestro algoritmo. Intentamos encontrar evidencias que permitan inferir si existe algun motivo lineal común a las proteínas que se incorporan al nucleolo.
Como aclaración importante, debemos hacer notar que las proteínas que tienen la capacidad de ingresar dentro del nucleolo tienen también la capacidad de ingresar dentro del núcleo celular desde el citoplasma. Por lo tanto, todo el conjunto de proteínas a estudiar comparte también esta característica. Se conoce que para ingresar dentro del núcleo, la secuencia que describe a las proteínas debe tener un motivo lineal compuesto por una secuencia de lisinas (K) y argininas (R) [Chelsky 1989]. Por lo tanto, es de esperarse que en las secuencias a ser analizadas se encuentren motivos de esta naturaleza.

Resultados

Se examinó la literatura especializada en busca de proteínas de localización nucleolar. A fin de facilitar la tarea del algoritmo, se tuvieron en cuenta proteínas para las que se ha acotado el segmento de secuencia mediador de la localización a 100 aminoácidos o menos. Se recopilaron 73 proteínas con estas características [Emmot 2009,Scott 2010] .

Los datos de entrada fueron analizados mediante el algoritmo, buscando descubrir motivos sobrerrepresentados. Al no conocer el largo de los motivos hipotéticos, se han ejecutado varias corridas del algoritmo con ventanas de 5, 7, 9, 11, 13 y 15 aminoácidos. Los resultados con ventanas de 11, 13 y 15 aminoácidos no difieren de los resultados obtenidos con una ventana de 9, por lo que analizaremos los resultados obtenidos para ventanas de 5, 7 y 9 aminoácidos.

Largo 5

Representamos aquí los alineamientos con mayor cantidad de información. Podemos observar que el primer resultado con mayor información de largo 5 es una secuencia de 5 lisinas, si bien la cuarta posición está peor determinada.

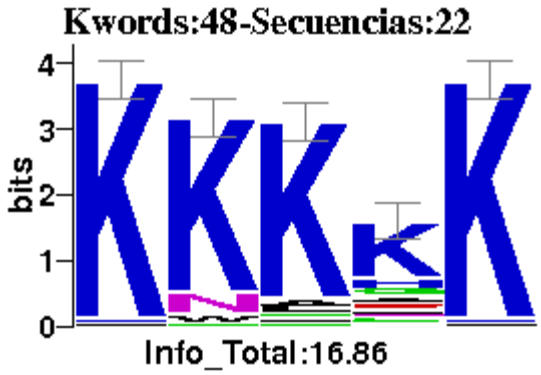


Figura 28: Motivo de largo 5 con mayor puntuación

Luego aparece un alineamiento donde la tercera posición tiene mayor probabilidad de ser Histidina (H), si bien se observan también apariciones también de lisina (K) y arginina (R).

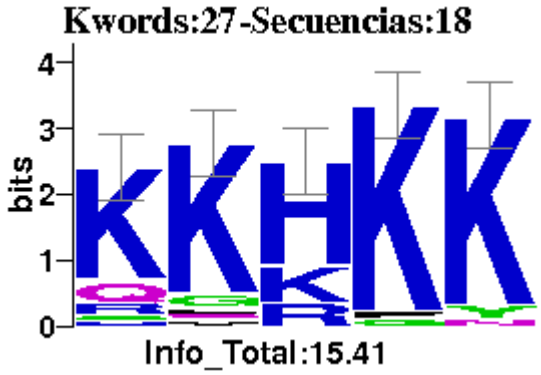


Figura 29: Motivo de largo 5 con segunda mayor puntuación

Posteriormente tenemos incertidumbre sobre los aminoácidos que ocupan las posiciones 2, 4 y 5.

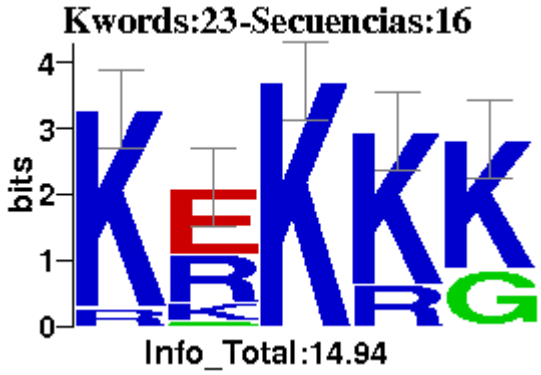


Figura 30: Motivo de largo 5 con tercer mayor puntuación

Por último encontramos un alineamiento que arroja buenas posibilidades de que exista en la tercera posición una Arginina (R).

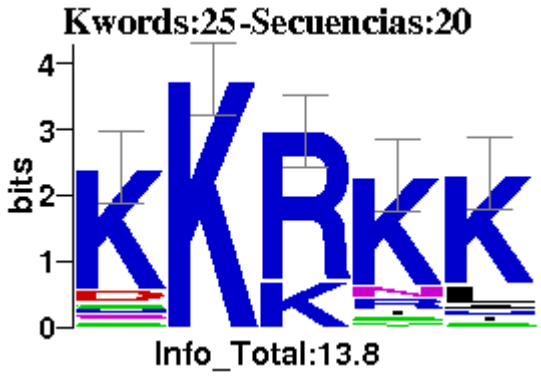


Figura 31: Motivo de largo 5 con cuarta mayor puntuación

Queremos destacar que el primer alineamiento no solamente es el de mayor contenido en información, sino el que agrupa k-words de un mayor número de secuencias. Esto lo hace más relevante, ya que propone un motivo lineal común a más secuencias de los datos de partida.

Respecto a los motivos encontrados, si bien están dominados por la presencia de lisinas (K), también encontramos posiciones dominadas por otros aminoácidos químicamente similares, como la arginina (R), la histidina (H) o el glutámico (E).

Largo 7

El alineamiento de mayor información presenta solamente cinco posiciones de alto contenido en información, mientras que las otras dos posiciones aparentan estar peor determinadas. Nuevamente predomina la presencia de lisinas.

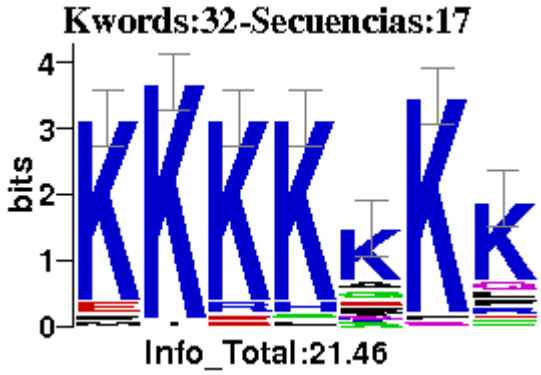


Figura 32: Motivo de largo 7 con mayor puntuación

Luego tenemos otro alineamiento con seis posiciones bien determinadas. Cinco de ellas son lisinas y una un glutámico o una lisina con casi la misma probabilidad.

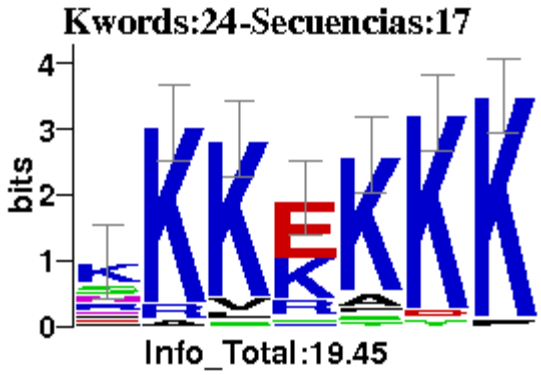


Figura 33: Motivo de largo 7 con segunda mayor puntuación

El alineamiento siguiente cuenta con siete posiciones bien definidas. Resulta interesante observar la presencia de argininas (R) en dos de ellas.

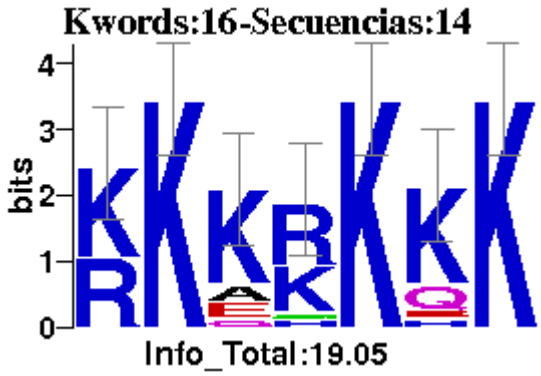


Figura 34: Motivo de largo 7 con tercer mayor puntuación

Con algo menos de información podemos tener un alineamiento con siete posiciones bien definidas, todas ellas dominadas por lisinas.

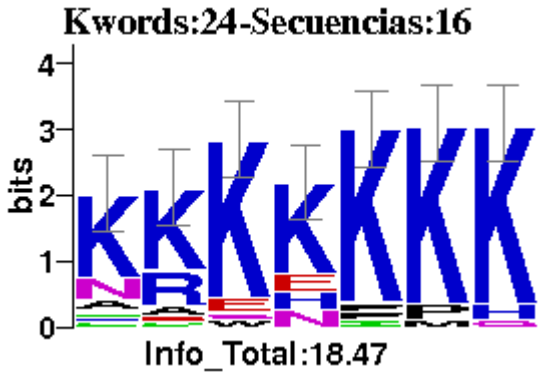


Figura 35: Motivo de largo 7 con cuarta mayor puntuación

Respecto a los alineamientos de largo cinco, observamos una composición similar en términos de aminoácidos. La mayor novedad es que encontramos posiciones sin una preferencia

definida, lo que no ocurría en los alineamientos de largo cinco. Nuevamente, la cantidad de secuencias representadas en los alineamientos (de 14 a 17) es baja respecto del número total de secuencias (73).

Largo 9

Vemos que al crecer el largo aumenta el contenido en información de los primeros alineamientos de la lista, aunque no de una forma radical. En paralelo, disminuye paulatinamente el número de secuencias representadas en dichos alineamientos.

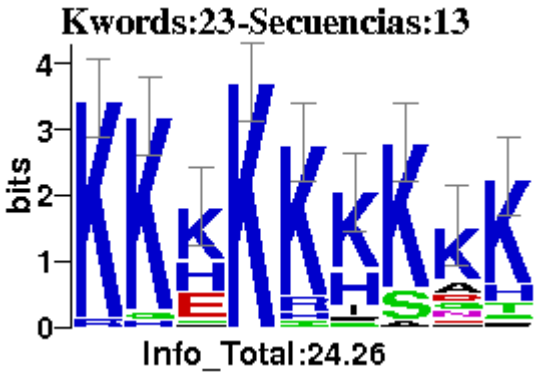


Figura 36: Motivo de largo 9 con mayor puntuación

También es notable la menor fracción de posiciones en las que sólo aparece la lisina.

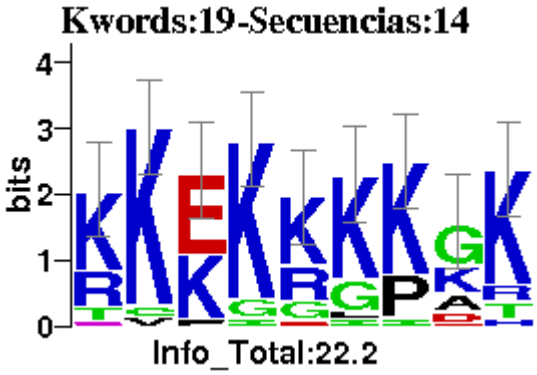


Figura 37: Motivo de largo 9 con segunda mayor puntuación

Sí se conserva la identidad de los aminoácidos dominantes en las representaciones gráficas: a la K la acompañan con frecuencia R, E y H.

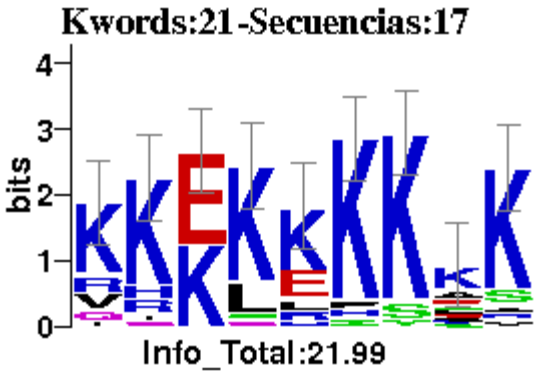


Figura 38: Motivo de largo 9 con tercermayor puntuación

Ventanas de largos mayores

Mostramos ahora el alineamiento de mayor información para una ventana de largo 15.

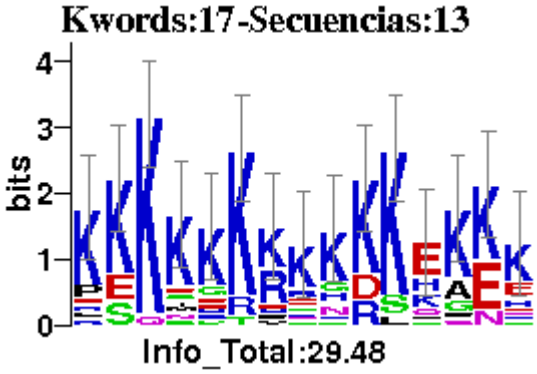


Figura 39: Motivo de largo 15 con mayor puntuación

La cantidad de información no se ha visto mayormente incrementada respecto de la ventana de largo nueve, considerando que el alineamiento tiene seis posiciones más. Por otro lado, vemos que muchas posiciones presentan un bajo contenido en información. Además, el alineamiento está conformado por k-words de solamente 13 secuencias, lo que indica que el motivo lineal representado sólo explicaría la localización nucleolar de una pequeña fracción de las proteínas consideradas en los datos de entrada.

Evolución de la información

Los siguientes gráficos muestran la evolución del contenido en información de los mejores cien alineamientos en función de la cantidad de rondas del algoritmo para largos de ventana cinco y siete. Recordemos que el algoritmo corre hasta el momento en que ningún alineamiento ve modificada su información. Aquí estamos visualizando únicamente los mejores cien alineamientos.

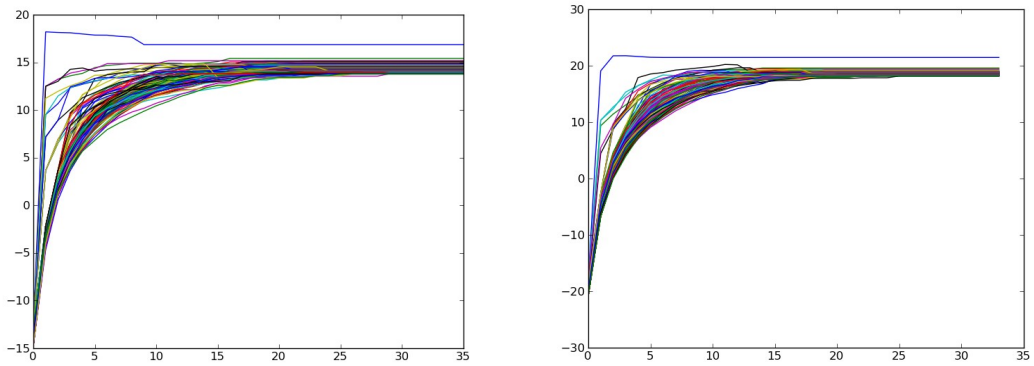


Figura 40: Evolución de la información en función de las rondas del algoritmo para largos 5 y 7.

Podemos observar que el contenido en información de la mayoría de los cien mejores alineamientos llega a su valor final en una etapa temprana del algoritmo.

Análisis de los resultados obtenidos

La característica principal de los resultados obtenidos es que para los largos cinco, siete y nueve pudimos identificar motivos sobrerrepresentados con posiciones que poseen una alta cantidad de información. Los motivos están compuestos mayormente por lisinas (K), aunque tienen diferentes variaciones entre sí que incluyen argininas (R), histidinas (H) y glutámicos (E). Otra característica importante es que ninguno de los motivos encontrados incluye la totalidad de las secuencias de entrada. Esto sugiere que las secuencias de entrada podrían responder a varios subconjuntos, cada uno asociado con un motivo lineal diferente.

Se conoce que el motivo lineal que permite a una proteína ingresar dentro del núcleo celular está compuesto por argininas y lisinas [Chelsky 1989]. Se ha propuesto el consenso K(KR)x(KR) para dicho motivo, con una longitud de cuatro residuos. Las proteínas que estamos estudiando deben ingresar al núcleo para poder incorporarse al nucleolo. Esto sugiere que los motivos lineales obtenidos podrían estar asociados al ingreso al núcleo y no a la incorporación al nucleolo. En este caso, el mayor contenido en información de los motivos de localización nuclear estaría dificultando la visualización de los motivos lineales asociados con la incorporación al nucleolo.

En etapas futuras el análisis aplicaremos nuestro algoritmo a proteínas que ingresan al núcleo celular pero no se incorporan al nucleolo. Esto nos permitirá comprobar si hay una correspondencia entre los motivos de localización nuclear y los aquí reportados. Una conjetura posible es que el motivo lineal que permite la incorporación al nucleolo se superpone con el motivo de localización nuclear. En ese caso, deberíamos observar diferencias sutiles entre los motivos de localización nuclear y los aquí reportados. Por ejemplo, el contenido en información debería ser mayor para los motivos de proteínas de nucleolo, ya que contienen dos motivos superpuestos.

Discusión, conclusiones y trabajo futuro

Modificaciones con respecto al algoritmo original para motivos lineales de ADN

Las modificaciones presentadas en nuestro desarrollo con respecto al algoritmo original presentado por Stormo explicado en la introducción del presente trabajo son las siguientes:

- Aplicación de filtros sobre la entrada, inspirados por los algoritmos existentes para motivos lineales de proteínas.
- Posibilidad de filtrar la salida mediante un análisis de correlación de matrices.
- Posibilidad de tratar a las k-words como si todas pertenecieran a un mismo conjunto, sin distinguirlas por pertenencia a secuencias. En otras palabras, si un motivo está sobrerrepresentado, no descartamos el hecho de que pueda estar presente más de una vez en cada secuencia. Esto último nos obliga a no agregar más de una vez una misma k-word o una k-word que se solape con alguna de las k-words que se tienen dentro del alineamiento que se está analizando.
- Presentación de los resultados utilizando herramientas probabilísticas basadas en teoría de la información.
- Presentación de la evolución del contenido en información en función de las rondas de ejecución.
- Detención del algoritmo únicamente cuando no exista la posibilidad de que ninguna k-word haga incrementar la información de alguno de los alineamientos que se están analizando.
- Precálculo de la información de cada alineamiento para acelerar los tiempos de corrida, como se propone en [Schneider 1996].

Cabe destacar que algunas de estas opciones de modificación son de aplicación opcional.

Comparación con algoritmos para motivos lineales de proteínas

Existen dos algoritmos principales para la búsqueda de nuevos motivos lineales funcionales en proteínas, Dilimot y SLiMFinder. Nuestro algoritmo toma de los algoritmos ya publicados la aplicación de filtros a las secuencias de entrada. En concreto, proporcionamos como opción el filtrar dominios globulares. Por otro lado, debemos señalar las siguientes diferencias:

- Nuestro algoritmo no filtra metioninas iniciales, secuencias transmembrana, regiones de baja complejidad, regiones de colágeno o péptidos de señalización. En nuestra opinión, la visualización de los resultados mediante logos de secuencia hace innecesario el filtrado.

- El filtro CD-HIT elimina secuencias parecidas entre sí usando un valor de umbral especificado por el usuario, haciendo posible filtrar redundancias debidas a la homología de las secuencias de entrada.
- A diferencia de Dilimot, nuestro algoritmo no filtra la salida teniendo en cuenta la conservación del motivo. Este cambio nos permite analizar la conservación del motivo propuesto, a la manera del trabajo de Dinkel y Sticht.
- A diferencia de SLiMFinder, no consideramos motivos de longitud variable.
- A diferencia tanto de SLiMFinder como de Dilimot, nuestra representación de los resultados es probabilística y no se ciñe a secuencias consenso definidas de manera estricta. Opinamos que esto profundiza nuestra comprensión de los motivos lineales considerados.
- Nuestro algoritmo puede agrupar los motivos obtenidos mediante un análisis de correlación. Esto permite al usuario identificar familias de motivos relacionados.

Conclusiones

1. Hemos implementado un nuevo método para buscar nuevos motivos lineales en proteínas.
2. El método heurísitico compara bien contra una versión exhaustiva y corre en tiempos mucho más cortos.
3. El método obtiene resultados similares al estándar del campo (Dilimot) en la búsqueda de motivos ya conocidos, inclusive mejorando el resultado obtenido en algunos casos.
4. Hemos aplicado el algoritmo a un nuevo caso de estudio, para el que hemos obtenido resultados preliminares.

Trabajo futuro

Es fundamental destacar que el presente trabajo es un desarrollo que comienza desde cero y excede el ámbito de esta tesis. Aquí enunciaremos algunas propuestas para trabajar sobre el método en el futuro.

Búsqueda automática de tamaño de ventana

Uno de los problemas que se presentan cuando se intentan descubrir motivos lineales y no se posee demasiada información sobre la interacción que se planea describir es que no se conoce el tamaño del motivo. En la aplicación de nuestro método esto resulta un problema que puede ser resuelto por el usuario de manera heurísitica mediante alguna de estas propuestas:

- Ejecutar el método para un tamaño de ventana grande y analizar la longitud de los motivos resultantes. Por ejemplo, podríamos buscar un motivo de largo diez y obtener como resultado el gráfico de la figura 37. Un análisis válido sería deducir que existe dentro del conjunto de datos un motivo de largo cuatro cuya descripción es KATR.

Figura 41: Ejemplo de una ventana de largo cuatro hallada en una corrida del algoritmo buscando motivos de largo 10.

- Ejecutar el método para varios largos de motivo distintos y analizar los resultados buscando cuál es el largo donde mejor promedio de información sobre posición se obtiene y cuáles son los motivos que se van repitiendo a lo largo de las diferentes corridas.

En el futuro, sería un progreso no solicitar al usuario que indique el largo de motivo que está buscando, o darle la posibilidad de que no lo especifique si es que aún no tiene suficiente información sobre la interacción que está analizando.

Validación con datos biológicos

En la sección de resultados ponemos a prueba la capacidad del algoritmo de encontrar 11 motivos conocidos. Se obtienen resultados buenos en 6 de ellos, resultados parciales en 2 de ellos y el método fracasa en 3 casos. Si bien estos números son esperanzadores, resulta recomendable incorporar más motivos a la validación en el futuro y realizar una exploración sistemática del efecto de todas las opciones de corrida en el desempeño del algoritmo.

En principio, el algoritmo debería ser capaz de detectar múltiples motivos presentes en un solo conjunto de secuencias de entrada. Pondremos a prueba dicha capacidad combinando varios conjuntos de secuencias correspondientes a motivos distintos usados en la validación primaria.

También examinaremos la degradación del desempeño del algoritmo al añadir al conjunto de secuencias de entrada (con un motivo en común) un número progresivamente mayor de secuencias tomadas al azar de la base de datos Uniprot. Este experimento mediría la capacidad del algoritmo de detectar un motivo en presencia de ruido.

Mejora del factor de corrección

Hemos mencionado en la sección introductoria de este trabajo que la forma en la que nosotros calculamos el factor de corrección de la información para un número discreto de muestras es aproximado. En el mismo trabajo en donde se menciona ese método aproximado [Schneider 1986] se describe también un método exacto para calcular dicho factor de corrección. Si bien el método exacto tiene un muy alto costo computacional, solamente sería necesario hacer los

cálculos una única vez. Una vez obtenidos, los resultados podrían incorporarse al algoritmo sin perjuicio de la velocidad de ejecución.

Implementación en otro lenguaje

El método fue desarrollado en el lenguaje Python por la rapidez que otorga a la hora de programar y por encontrarnos en una etapa inicial de la aplicación del método. En el futuro, sería positivo analizar la conveniencia de volver a implementar el algoritmo utilizando un lenguaje de programación que permita acelerar las ejecuciones y, sobre todo, que otorgue escalabilidad al posible crecimiento del desarrollo.

Procesamiento paralelo

Observamos que las limitaciones computacionales del método en su versión actual vienen dadas por la capacidad de procesamiento y no por las capacidades de memoria. Por lo tanto sería una buena decisión analizar la mejor manera de realizar cómputos paralelos al ejecutar el método y cómo eso influiría en su agilización. De implementarse este inciso, sería conveniente verificar las mejoras en la ejecución mediante pruebas en un cluster.

Entorno de ejecución

El presente trabajo dio como fruto un algoritmo del cual el usuario ejecuta corridas individuales, que después analiza. Sería útil poner a disposición del usuario un entorno de ejecución en donde puedan salvarse resultados parciales o finales y explorar variantes en el análisis de los resultados. Enumeramos algunas hipotéticas situaciones que nos gustaría poder reproducir:

- En una ronda determinada, frenar la ejecución del algoritmo y agrupar mediante el análisis de correlación los alineamientos actuales.
- Serializar y guardar en un archivo una matriz de resultados para su posterior análisis y para que éste pueda ser realizado por varias personas de manera separada.
- Definir interactivamente el tamaño de ventana en función de los resultados parciales que va arrojando el algoritmo.
- Establecer un umbral que permita descartar los motivos con menor cantidad de información en determinada ronda del algoritmo, lo que permitiría achicar los tiempos de ejecución.

Servidor Web

Para una validación más amplia del método y su posterior aplicación sería bueno disponer de un servidor web en donde éste se encuentre publicado y al que pueda acceder la comunidad [5,46]. Antes de llevar a cabo esto, sería fundamental discutir el entorno de ejecución y la forma de presentación de los datos, ya que cada corrida genera una gran cantidad de información diseminada en diferentes archivos. También sería deseable disminuir los tiempos típicos de corrida, que son de varias horas en la implementación actual.

Bibliografia

- **Aasland 2002:** FEBS Lett. 2002 Feb 20;513(1):141-4. Normalization of nomenclature for peptide motifs as ligands of modular protein domains. Aasland R, Abrams C, Ampe C, Ball LJ, Bedford MT, Cesareni G, Gimona M, Hurley JH, Jarchau T, Lehto VP, Lemmon MA, Linding, Mayer BJ, Nagai M, Sudol M, Walter U,Winder SJ.
- **Aloy 2005:** Curr Opin Struct Biol. 2005 Feb;15(1):15-22. Protein complexes: structure prediction challenges for the 21st century. Aloy P, Pichaud M, Russell RB.
- **Bailey 1951:** Annu Rev Biochem. 1951;20:103-30. The chemistry of amino acids and proteins. Bailey K, Sanger F.
- **Bassi 2007:** PLoS Comput Biol. 2007 Nov;3(11):e199. A primer on python for life science researchers. Bassi S.
- **Borysko 1951:** Bull Johns Hopkins Hosp. 1951 Dec;89(6):468-73. Structure of the nucleolus as revealed by the electron microscope; a preliminary report. Borysko E, Bang FB.
- **Cock 2009:** Bioinformatics. 2009 Jun 1;25(11):1422-3. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Cock PJ, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B, de Hoon MJ.
- **Crooks 2004:** Genome Res. 2004 Jun;14(6):1188-90. WebLogo: a sequence logo generator. Crooks GE, Hon G, Chandonia JM, Brenner SE.
- **Chelsky 1989:** Mol Cell Biol. 1989 Jun;9(6):2487-92. Sequence requirements for synthetic peptide-mediated translocation to the nucleus. Chelsky D, Ralph R, Jonak G.
- **Chemes 2010:** Febs J. 2010;277(4):973-88. Targeting mechanism of the retinoblastoma tumor suppressor by a prototypical viral oncoprotein. Structural modularity, intrinsic disorder and phosphorylation of human papillomavirus E7. Chemes LB, Sánchez IE, Smal C, de Prat-Gay G.
- **Dinkel 2007:** A computational strategy for the prediction of functional linear peptide motifs in proteins, Dinkel H, Sticht H - 2007
- **Edwards 2007:** SLiMFinder: A probabilistic Mehod for Identifying Over-Represented Convergently Envolved, Short Linear Motifs in Proteins. PLoS ONE 2(10): e967. Edwards RJ, Davey NE, Shields DC - 2007
- **Emmot 2009:** EMBO Rep. 2009 Mar;10(3):231-8. Nucleolar targeting: the hub of the matter. Emmott E, Hiscox JA.
- **Encinar 2009:** Bioinformatics. 2009 Sep 15;25(18):2418-24. ADAN: a database for prediction of protein-protein interaction of modular domains mediated by linear motifs. Encinar JA, Fernandez-Ballester G, Sánchez IE, Hurtado-Gomez E, Stricher F, Beltrao P, Serrano L.
- **Gould 2010:** Nucleic Acids Res. 2010 Jan;38(Database issue):D167-80. ELM: the status of the 2010 eukaryotic linear motif resource. Gould CM, Diella F, Via A, Puntervoll P, Gemünd C, Chabanis-Davidson S, Michael S, Sayadi A, Bryne JC, Chica C, Seiler M,

Davey NE, Haslam N, Weatheritt RJ, Budd A,Hughes T, Pas J, Rychlewski L, Travé G, Aasland R, Helmer-Citterich M, Linding R, Gibson TJ.

- **Han 2004:** Nature. 2004 Jul 1;430(6995):88-93. Evidence for dynamically organized modularity in the yeast protein-protein interaction network. Han JD, Bertin N, Hao T, Goldberg DS, Berriz GF, Zhang LV, Dupuy D, Walhout AJ, Cusick ME, Roth FP, Vidal M.
- **Magrane 2011:** Database (Oxford). 2011 Mar 29;2011:bar009. Print 2011. UniProt Knowledgebase: a hub of integrated protein data. Magrane M, Consortium U.
- **Neduva 2005:** "Systematic Discovery of New Recognition Peptides Mediating Protein Interaction Networks" Neduva V, Linding R, Su-Angrand I, Stark A, Masi Fd, et al. (2005). PLoS Biol 3(12): e405.
- **Neduva 2006:** Nucleic Acids Res. 2006 Jul 1;34(Web Server issue):W350-5. DILIMOT: discovery of linear motifs in proteins.Neduva V, Russell RB.
- **Oppel 1960:** Ukr Biokhim Zh. 1960;32:742-69. Primary structure of the globin component in hemoglobin. Oppel VV.
- **Pearson 1988:** Proc Natl Acad Sci U S A. 1988 Apr;85(8):2444-8. Improved tools for biological sequence comparison. Pearson WR, Lipman DJ.
- **Perutz 1960:** Nature. 1960 Feb 13;185(4711):416-22. Structure of haemoglobin: a three-dimensional Fourier synthesis at 5.5-A. resolution, obtained by X-ray analysis. Perutz MF, Rossmann MG, Cullis AF, Muirhead H, Will G, North AC.
- **Puntervoll 2003:** Nucleic Acids Res. 2003 Jul 1;31(13):3625-30. ELM server: A new resource for investigating short functional sites in modular eukaryotic proteins. Puntervoll P, Linding R, Gemünd C, Chabanis-Davidson S, Mattingsdal M, Cameron S, Martin DM, Ausiello G, Brannetti B, Costantini A, Ferrè F, Maselli V, Via A, Cesareni G,Diella F, Superti-Furga G, Wyrwicz L, Ramu C, McGuigan C, Gudavalli R, Letunic I, Bork P, Rychlewski L, Küster B, Helmer-Citterich M, Hunter WN, Aasland R, Gibson TJ.
- **Rastogi 2010:** Methods Mol Biol. 2010;619:285-305. Bioinformatics predictions of localization and targeting. Rastogi S, Rost B.
- **Rinderknecht 1976:** Proc Natl Acad Sci U S A. 1976 Dec;73(12):4379-81.Amino-terminal sequences of two polypeptides from human serum with nonsuppressible insulin-like and cell-growth-promoting activities: evidence for structural homology with insulin B chain. Rinderknecht E, Humbel RE.
- **Rodgers 1988:** J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. The American Statistician, 42(1):59–66, February 1988
- **Scott 2009:** Science. 2009 Nov 27;326(5957):1220-4. Cell signaling in space and time: where proteins come together and when they're apart. Scott JD, Pawson T.
- **Scott 2010:** Nucleic Acids Res. 2010 Nov 1;38(21):7388-99. Characterization and prediction of protein nucleolar localization sequences. Scott MS, Boisvert FM, McDowall MD, Lamond AI, Barton GJ.
- **Schneider 1986:** J Mol Biol. 1986 Apr 5;188(3):415-31. Information content of binding sites on nucleotide sequences. Schneider TD, Stormo GD, Gold L, Ehrenfeucht A.
- **Schneider 1990:** Nucleic Acids Res. 1990 Oct 25;18(20):6097-100. Sequence logos: a new way to display consensus sequences. Schneider TD, Stephens RM.

- **Schneider 1991:** J Theor Biol. 1991 Jan 7;148(1):83-123. Theory of molecular machines. I. Channel capacity of molecular machines. Schneider TD.
- **Schneider 1991:** J Theor Biol. 1991 Jan 7;148(1):125-37. Theory of molecular machines. II. Energy dissipation from molecular machines. Schneider TD.
- **Schneider 1996:** Discrete Appl Math. 1996 Dec 1;71(1-3):259-268. Fast multiple alignment of ungapped DNA sequences using information theory and a relaxation method. Schneider TD, Mastronarde DN.
- **Schneider 1997:** "Information content of individual genetic sequences" Schneider TD. J Theor Biol. 1997 Dec 21;189(4):427-41.
- **Schneider 2001:** Nucleic Acids Res. 2001 Dec 1;29(23):4881-91. Strong minor groove base conservation in sequence logos implies DNA distortion or base flipping during replication and transcription initiation. Schneider TD.
- **Schneider 2010:** Appl Bioinformatics. 2002;1(3):111-9. Consensus sequence Zen. Schneider TD.
- **Schneider 2010:** Nucleic Acids Res. 2010 Oct;38(18):5995-6. 70% efficiency of bistate molecular machines explained by information theory, high dimensional geometry and evolutionary convergence. Schneider TD.
- **Stormo 1989:** "Identifying protein-binding sites from unaligned DNA fragments" Stormo GD, Hartzell GW 3rd. Proc Natl Acad Sci U S A. 1989 Feb;86(4):1183-7.
- **Stormo 1989:** Proc Natl Acad Sci U S A. 1989 Feb;86(4):1183-7. Identifying protein-binding sites from unaligned DNA fragments. Stormo GD, Hartzell GW 3rd.
- **Uversky 2010:** Biochim Biophys Acta. 2010 Jun;1804(6):1231-64. Understanding protein non-folding. Uversky VN, Dunker AK.
- **Vucetic 2003:** Vucetic S., Brown C.J., Dunker A.K. and Obradovic Z., Flavors of protein disorder (2003). Proteins 52 (4); 573-584.
- **Weizhong 2006:** Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences", Weizhong Li & Adam Godzik Bioinformatics, (2006) 22:1658-9
- **Xue 2010:** Biochim Biophys Acta. 2010 Apr;1804(4):996-1010. PONDR-FIT: a meta-predictor of intrinsically disordered amino acids. Xue B, Dunbrack RL, Williams RW, Dunker AK, Uversky VN.