Detección de proteínas multifuncionales con funciones celulares ocultas utilizando BIG-DATA.

Helmut Enöckl Cabacas

Resumen— Las proteínas moonlighting son proteínas con funciones múltiples totalmente independientes y ocultas, es decir, se desconocen muchas de las funciones que tienen las proteínas moonlighting. Además, las proteínas moonlitghting están estrechamente ligadas a enfermedades genéticas.

En este articulo se han descubierto nuevas funciones desconocidas en las proteínas de una forma inédita, ya que, hasta el momento, las que se han descubierto han sido, principalmente, en laboratorio. Y, en este trabajo, se ha realizado el proceso de detección de nuevas funciones desconocidas desde la bioinformática. Para ello, se recoge información en lenguaje natural de las bases de datos *Gene Ontology* [1], funciones asociadas a los genes, y *UniProt* [2], funciones asociadas a las proteínas.

La base de datos de Gene Ontology fue creada para tener toda la información de todos los genes y la base de datos de Uniprot para tener toda la información sobre las proteínas. En este trabajo utilizamos los datos que están relacionados con el Homo sapiens y usamos las bases de datos para descubrir funciones desconocidas en las proteínas mediante un algoritmo de árbol de decisiones basado en reglas biológicas.

El descubrimiento de las nuevas funciones desconocidas puede permitir saber las aflicciones que sufren las proteínas, conocer reacciones desconocidas, al entrar en contacto con algún medicamento o la relación con algunas patologías.

Palabras clave— UniProt, Geneontology, proteínas moonlighting, Big-Data, árbol de decisiones.

Abstract— Moonlighting proteins are proteins with multiple totally independent and hidden functions and, that is to say, that many of the functions that have the moonlighting proteins are unknown. They are also closely linked to genetic diseases.

This Article pretends to discover new hidden functions in proteins in an unpublished way, since those that have been discovered have been by laboratory and now, the process of detecting new unknown functions has been performed from bioinformatics. For it collects information in natural language from database, *Gene Ontology* [1], functions associated to genes, and the database *Uniprot* [2], functions associated to protein.

The Gene Ontology database was created to have all the information of all the genes and the Uniprot database to have all the information about the proteins. In this article, we use data that are related to Homo sapiens and we use databases to discover unknown functions in proteins using a decision tree algorithm based on biological rules.

The discovery of the new unknown functions can allow to know the afflictions that suffer the proteins when coming in contact with some medicine or the relation with some pathologies.

Index Terms— Uniprot, Geneontology, moonlighting proteins, Big-Data, decision tree.

1. Introducción

Las proteínas moonlighting son proteínas con funciones múltiples totalmente independientes. La identificación de proteínas multifuncionales es muy importante por muchas razones: en primer lugar, estas pueden tener un papel regulador en su función no canónica. Además, están muy relacionadas con patologías. Un 76% de las proteínas moonlighting están relacionadas con enfermedades genéticas descritas en la base de datos OMIM. Entre los targets farmacéuticos conocidos, un 47% actuaría sobre proteínas moonlighting. Algunas proteínas moonlighting [3] realizan su función extra sólo en células tumorales y son importantes para la progresión de la enfermedad. Por último, una función extra desconocida de una proteína puede causar efectos secundarios inesperados en un fármaco que actúe sobre esta proteína.

E-mail de contacto: Helmut.enockl@e-campus.uab.cat
Mención realizada: Inginería de Computación.
Trabajo tutoritzado por: Mario Huerta
Curso 2016/17

Las proteínas multifuncionales que hasta ahora han sido identificadas fueron descubrimientos fortuitos y son probablemente una fracción pequeña de la totalidad. La predicción a gran escala de proteínas multifuncionales, sin embargo, requiere el desarrollo de herramientas específicas.

Este trabajo ha realizado la detección de proteínas moonlighting con funciones no conocidas de la proteína moonlighting, a partir del data minning, técnica utilizada para descubrir patrones sobre bases de datos. En mi caso, lo utilice sobre la base de datos *Gene Ontology*, base de datos de funciones asociadas a los genes, y *Uniprot*, base de datos de funciones asociadas a las proteínas.

La base de datos *Gene Ontology*, contiene los genes recogidos por National Human Genome Research Institute (NHGRI) para *Gene Ontology Consortium*, y *UniProt*, base de datos de proteínas de European Bioinformatics Institute.

Diferentes funciones son asociadas a las proteínas a partir de experimentos que diferentes grupos de investigadores realizan en todo el mundo. En la base de datos *Gene Ontology* estas funciones son anotadas a la correspondiente proteína. El problema es que muchas de estas funciones anotadas forman parte de la misma función o proceso y no representan realmente funciones independientes de la proteína.

Por eso, es necesario discernir entre las diferentes anotaciones de la base de datos.

Disponemos los siguientes conceptos clave:

- **GO:activity:** Actividades funcionales de la proteína.
- **GO:binding:** Funciones de acoplamiento de la proteína, donde la diferencia respecto las *GO:activity* es que los binding son términos para describir la función de la proteína de una forma muy genérica, pues solo dicen a que término se unen, y no para qué.
- **GO:function:** Conjunto de *GO:activitys* y *GO:bindings*.
- **GO:process:** Procesos celulares de la proteína.

Necesitamos conocer si *GO:process*, *GO:activitys*, *y GO:bindings* forman parte de una misma función o si representan funciones independientes, por lo que habríamos encontrado una nueva función desconocida.

2. Estado del arte

Actualmente existen algunos listados de proteínas moonlighting bastante limitados, ya que actualmente todos los métodos que se han realizado para detectar si una proteína tiene una función desconocida han sido a base de realizar ensayos en laboratorio, por lo que algunas de las funciones desconocidas que puede tener una proteína pueden pasar desapercibidas además de la lentitud y costes de realizar las pruebas en laboratorio.

No obstante, se han realizado varias investigaciones y pruebas sobre la aplicación de la Bioinformática para la detección de proteínas moonlighting. Pero ninguna de las investigaciones hasta el momento han intentado utilizar las bases de datos para descubrir proteínas multifuncionales.

Se realizó un estudio en el artículo "*Can bioinformatics help in the identification of moonlighting proteins?*"[4], donde se llegó a la conclusión que sí que sería posible.

Este trabajo recoge el inicio de las primeras impresiones tras aplicar la bioinformática y las bases de datos de genes y proteínas en el ámbito de proteínas multifuncionales.

3. Objetivo

El objetivo es crear un detector el cual sea capaz de descubrir nuevas funciones extra de las proteínas mediante data mining aplicado sobre los datos extraídos de las bases de datos mundiales: *Gene Ontology* y *UniProt*.

El descubrimiento de las nuevas funciones extra permitirá saber las aflicciones que sufren las proteínas al entrar en contacto con algún medicamento o la relación de éstas con algunas patologías. Esto contribuirá a la hora de la creación de nuevos fármacos que al disponer de más información sobre las posibles interacciones que pueden producirse en una proteína en concreto, la cual hasta el momento se pensaba que no se veía afectada.

4. Metodología

En esta sección se describen los algorítmos y ficheros utilizados para desarrollo del proyecto.

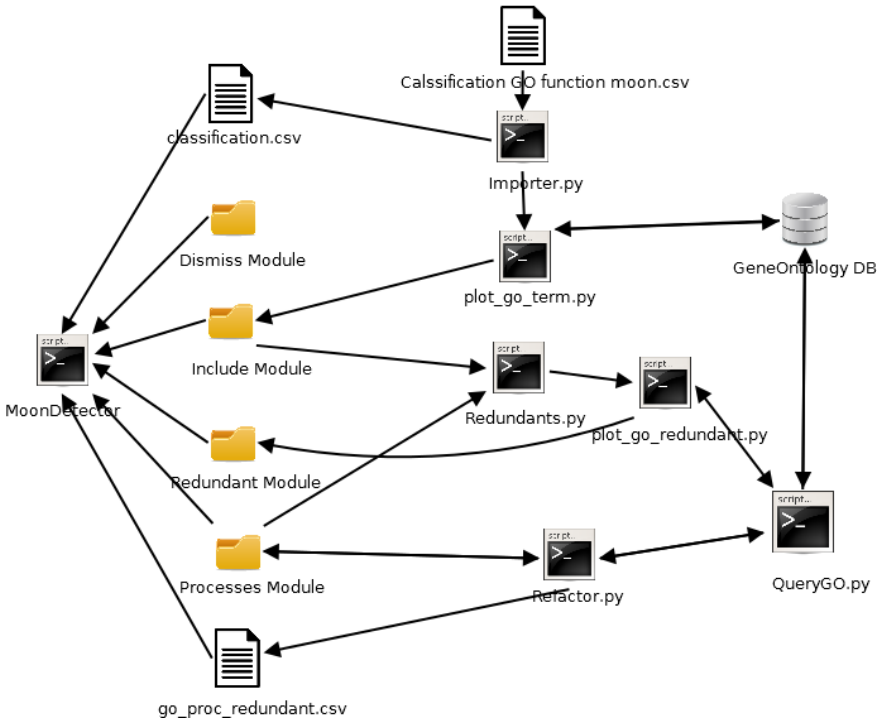

Figura 1: Diagrama de flujo que muestra las interacciones entre los distintos elementos que crean los diccionarios que utiliza *moonDetector*. Se pueden observar los scripts que participan en la creación de los diccionarios, esta acción se realiza en background. Y el script de *moonDetector* con los diccionarios, ya construidos, los utiliza para realizar la detección de las proteínas moonlighting.

Como se puede observar, en la *figura 1* el proceso es bastante costoso, ya que depende de muchas acciones y, por ello, la creación de los diccionarios se genera en un preproceso.

Los módulos se generan en 2 días, pero en el caso del conjunto de redundantes puede durar más de 3 meses en generar la totalidad de redundantes. Las bases de datos se actualizan mensualmente, como se pretende tener el conjunto de redundantes actualizado este proceso al terminar volverá a ejecutarse para recalcular los redundantes. No obstante, siempre se utilizarán los últimos redundantes calculados.

4.1. Módulos

- **Módulo include:** Ha de contener el diccionario de los procesos celulares y actividad funcionales. Hijos de un GO ID, identificador que se utiliza en la base de datos de *Gene Ontology* para identificar términos, consta de la palabra "GO:" seguido de 9 dígitos.

- **Módulo processes:** Ha de contener el diccionario de las funciones bindings.

- **Módulo redundant:** Este módulo ha de contener el diccionario de procesos celulares y actividades funcionales redundantes, es decir, que aparecen en la mayoría de genes.

- **Módulo dismiss:** Este módulo ha de contener el diccionario de procesos celulares, actividades funcionales o bindings a descartar.

- **Módulo detector:** Este módulo utilizará los diccionarios generados por el resto de módulos, para poder clasificar si una proteína es moonlighting mediante un árbol de decisiones.

- **Módulo de proteína:** Este módulo tiene que obtener los diccionarios de funciones y procesos asociados a una proteína. Sobre la base de datos *UniProt* extraeremos el código de gen y con *Gene Ontology*, junto el código de gen, extraemos los datos de los términos.

- **Módulo de testing:** Este módulo realizará comprobaciones mediante el método de deconstrucción del programa, ya que no se puede verificar con datos existentes.

  Con el método de reconstrucción pretendo hacer programas pequeños que verifiquen que los resultados extraídos son realistas con el comportamiento que ha de tener el detector con los diccionarios de los distintos módulos. Para ello, se realizarán pruebas con un conjunto de datos distinto, ya que no estará basado en pruebas biológicas.

4.2. Algoritmos y programas

- **getTerm:** Algoritmo que consulta la base de datos de *Gene Ontology* y devuelve el nombre del proceso celular, actividad funcional o binding dado una GO ID.

- **getGO:** Algoritmo que consulta la base de datos de *Gene Ontology* y devuelve el GO ID dado el nombre del proceso celular, actividad funcional o binding.

- **getTotalGenes:** Algoritmo que consulta la base de datos de *Gene Ontology* y devuelve el número de genes dado un GO ID.

- **Importer:** Lee un archivo csv, que contiene las actividades funcionales ya agrupadas, genera un fichero de clasificación. También, por cada función leída, llamará al *Plot_go_term*. Si la función no existe en ningún fichero del diccionario include generado hasta el momento, generará el árbol inferior y obtendrá el diccionario de la función.

  Además, también se extrae las funciones binding redundantes, con las que iremos generando un fichero con el nombre de la primera función no redundante (equivalente a la primera función extraída del paso anterior) junto todas las redundantes de esta función.

- **Refactor:** Lee cada fichero del módulo process que no contiene el identificativo GO y genera el mismo fichero, pero añadiendo el identificador, debido a que se iría llamando por cada nombre encontrado a la función *getGO*.

- **Plot_go_redundant:** Algoritmo que consulta la base de datos de *Gene Ontology* y que dado un GO ID, la tasa de redundancia y la extensión del fichero genera un fichero con todos los términos redundantes que tiene y que su tasa de repetición sea igual o superior a la introducida.

  Para ello, obtiene todos los genes asociados al GO ID y luego realiza otra consulta para obtener los términos que contienen estos genes y el número de veces que aparecen en ellos. Después dividimos el número de apariciones con el número de genes y sacamos la ratio de redundancia tal y como se muestra en la *fórmula 1* que se muestra a continuación:

$$\text{Tasa de redundancia} = \frac{\text{Número de apariciones}}{\text{Número de de genes}} \quad (1)$$

- **getGoTermsAllGenes:** Algoritmo que consulta la base de datos de *Gene Ontology* que, dado un código de gen e indicando el tipo de término que se desea, funciones o procesos, devuelve todos los términos asociados a ese gen y del tipo especificado.

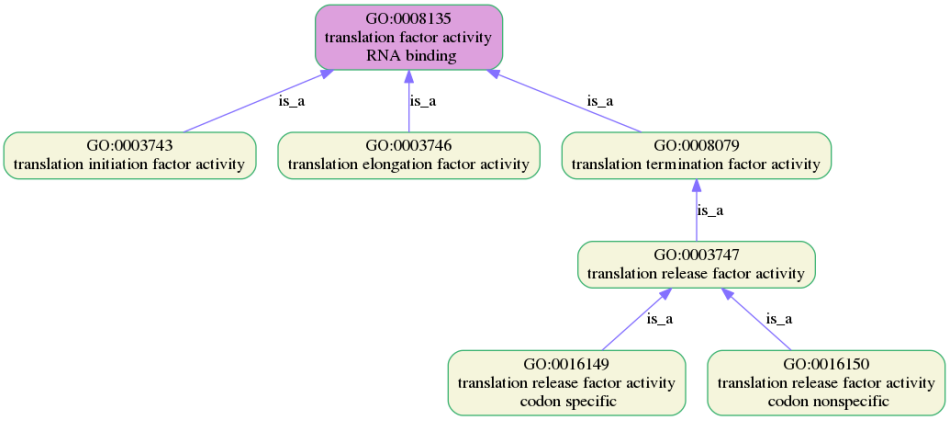

Figura 2: Árbol inferior del término GO 0008135

- **Plot_go_term:** Genera un fichero con el árbol inferior dado un proceso celular, actividad funcional o binding, como podéis ver en la *figura 2*. Con ello sabremos los hijos que tiene.

  Para realizarlo, se ha modificado la API de *Goatools 0.6.10[5],* herramienta para la búsqueda de información de genes, ya que generaba una imagen con el árbol completo y solo contenía la identificación del proceso celular, actividad funcional o binding, por lo que hemos utilizado la función *getTerm* para obtener además el nombre.

- **Plot_go_redundant_verif:** Variante del *Plot_go_redundant* y que es utilizado cuando las bases de datos no soportan consultas tan grandes. Este algoritmo que consulta la base de datos de *Gene Ontology* y que dado un GO ID, la tasa de redundancia y la extensión del fichero genera un fichero con todos los términos redundantes que tiene y que su tasa de repetición sea igual o superior a la introducida.

  Para ello, obtiene todos los genes asociados al GO ID y luego va realizando consulta de 1000 en 1000 genes o según este configurado, para obtener los términos que contienen estos genes y el número de veces que aparecen en ellos. Al finalizar, juntaría los resultados y después dividimos el número de apariciones con el número de genes y sacamos la ratio de redundancia.

- **Redundants:** Algoritmo que recorre una lista de procesos celulares y actividades funcionales para obtener sus redundantes. Por cada elemento de la lista lanza un thread que ejecuta *plot_go_Term* pasándole la GO ID, la tasa de redundancia y la

extensión. Si la GO ID está en el módulo de processes la extensión que pasará será "*gop*" o, por lo contrario, en el módulo de include será "*gof*".

- **FusionSimilar:** Este algoritmo pretende agrupar los términos que pertenecen a la misma familia. Para ello, dado un fichero csv de clasificación de proceso fusiona las líneas detectando los procesos que deben ir al mismo grupo utiliza búsquedas por cada palabra que hay en cada término de cada línea (se clasifica en forma de línea) si hay coincidencias, como mínimo de 3 palabras, sin tener en cuenta palabras a ignorar (muy comunes) o que al menos el número de palabras con coincidencia sea igual al tamaño del número de palabras del término más pequeño, sin tener en cuenta las palabras a ignorar, de los que se están comparando.

  De esta forma podremos crear atractores que funcionan como semillas y así poder juntar los términos más largos a estas semillas.

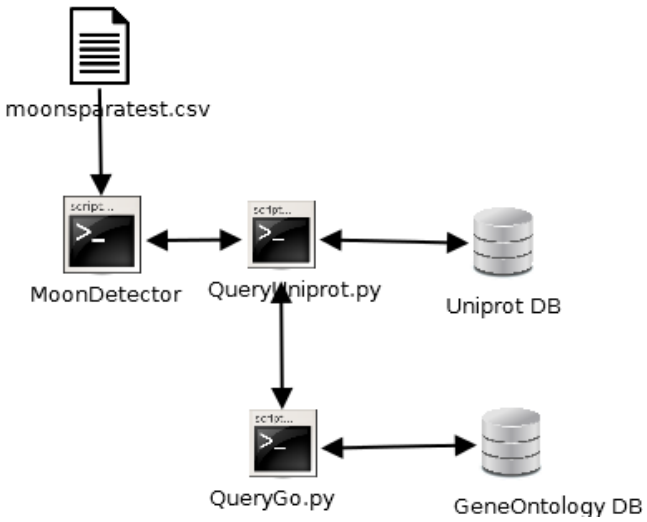

Figura 3: Esquema representativo de la obtención automática de actividades funcionales, funciones binding y los procesos ligados a proteínas dado un fichero con un listado de proteínas, en este caso *moonsparatest.csv*.

- **moonDetector:** Es el algoritmo de detección de proteínas moonligthing, como parámetro de entrada puede recibir un listado de proteínas representadas en la *figura 3* como *moonsparatest.csv* y el detector se encargará de extraer las actividades funcionales, las funciones binding y los procesos por cada proteína que este en el fichero. Además, el modulo también puede recibir de forma manual los datos de la proteína y no tener que calcularlo.

  El algoritmo de clasificación recibe una proteína con sus funciones y procesos. Entonces, por cada función verifica si es una binding, utilizando el nombre de la función, ya que contendrá la palabra binding.

  Utilizando los diccionarios de procesos, funciones y redundantes, y los dos ficheros de clasificación de funciones y procesos realizará lo siguiente por cada función:

  - Si es una binding intentará descartarla verificando si es redundante. Si no es redundante, tratará de asociarla a un proceso. En el caso de no poder, pondrá la función como función extra.

  - Si no es un binding intentará clasificar la función dentro de un grupo de funciones. Si no se puede clasificar mirará que no sea redundante e intentará clasificar dentro de un proceso. Si no es posible asignará la función como función extra.

  Una vez todas las funciones de la proteína se hayan verificado, se realiza una última comprobación para depurar las asignaciones en clasificación de funciones extra, clasificadas y procesos. Además, no permitirá que exista un mismo elemento en más de una lista.

  Asimismo, al finalizar todas las operaciones se realiza un post proceso con los resultados de procesos para agruparlos dando prioridad a los datos biológicos. Por ello, los separamos en 2 grupos los más importantes llamados antagonistas que son los que aparecen en un mismo grupo según los datos biológicos y los procesos que no aparezcan crearían la segunda lista de procesos extra.

  Por último, el algoritmo informará de los resultados e indicará que es una proteína extra, si existe más de una función clasificada, si solo hay una función clasificada, pero al menos existe una función extra o proceso con sus funciones asociadas, o bien si existe más de un proceso.

  Como podéis ver solo se contabiliza como una las funciones extra a diferencia de las funciones clasificadas o los procesos con sus asignaciones.

- **getSymbol:** Algoritmo que consulta la base de datos de UniProt y que una vez dado un código de proteína se encarga de facilitar el symbol del gen asociado.

- **getDataGOAllGenes:** Algoritmo que consulta la base de datos de *UniProt* y que dado un código de

proteína realiza una extracción de todos los términos y construye el diccionario de funciones y procesos asociados a la proteína.

3.3. Ficheros

- **Plot_go_term.py:** Contiene el algoritmo *plot_go_term*.
- **Plot_go_redundant.py:** Contiene el algoritmo *plot_go_redundant*.
- **Redundants.py:** Contiene el algoritmo *redundants*.
- **Import.py:** Contiene el algoritmo *import*.
- **Refactor.py**: Contiene el algoritmo *refactor*.
- **QueryGO:** Contiene las funciones *getTerm*, *getGO*, *getTotalGenes* y *getGoTermsAllGenes*.
- **QueryUniprot:** Contiene las funciones *getDataGOAllGenes* y *getSymbol*.
- **Ficheros csv:**
    1. Clasificacion GO funciones moon.
    2. Clasificacion GO Molecular Function.
    3. Molecular function 2 biological process.
    4. Moonsparatest.

4. RESULTADOS

Con el detector obtenemos unos resultados que indican si la proteína es moonlighting con las funciones clasificadas (*GO:activity*), procesos con sus funciones asociadas (*GO:process-GO:binding*) y las funciones extra (*unknown function*).

Primero, se realiza un preproceso para calcular los diccionarios. Una vez calculados los diccionarios, se guardan.

Con todos los diccionarios calculados ya se podría ejecutar de modo online el detector.

El detector recibe un listado de proteínas como entrada y extrae de las bases de datos *Gene Ontology* y *UniProt* un listado de *GO:function*, es decir, un conjunto de *GO:activitys* y *GO:bindings*, y un listado de *GO:process*, asociados a cada proteína. El detector trabaja a nivel individual cada proteína con sus *GO:function* y sus *GO:process*.

Ademas, también es posible introducir manualmente los *GO:function* y *GO:process* para realizar pruebas en este caso el detector trabajaría con los datos introducidos manualmente y no cargaría la información de las bases de datos *Gene Ontology* y *UniProt*.

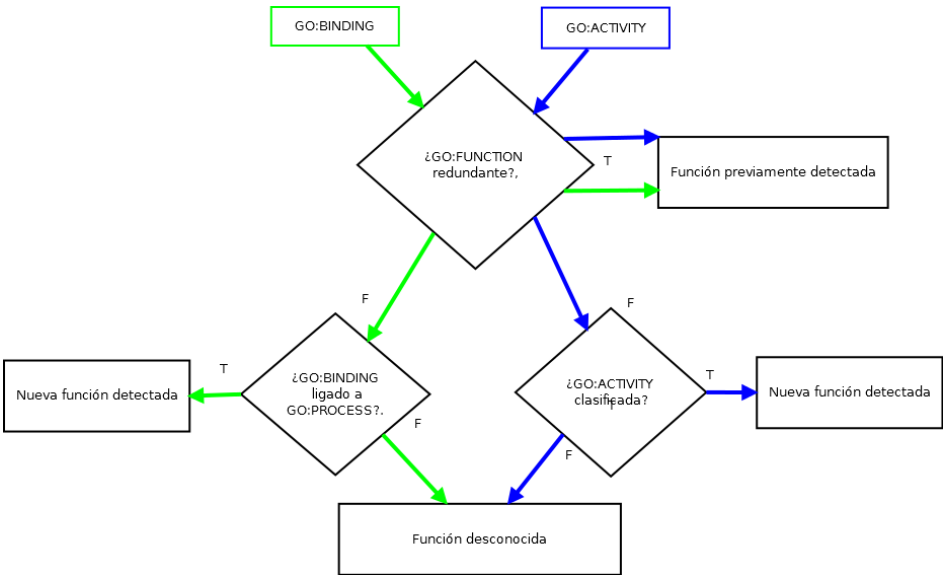

Figura 4: Esquema que describe el proceso de clasificación de las funciones.

En la *figura 4* se puede observar el modelo basado en procesos biológicos con la que se ha realizado la clasificación de funciones con lo que podemos verificar que el proceso es correcto. El algoritmo recibe como entrada una *GO:binding* o *una GO:activity* e intenta clasificarla como "función previamente detectada", "nueva función detectada" o función desconocida".

El detector para poder realizar la clasificación utiliza los árboles de términos de las activitys. Los árboles abarcan todos los términos que contiene la actividad funcional.

Por otro lado, los *GO:process*, además de ser un listado demasiado grande para ser tratado, no van ligados a una sola proteína, puesto que en un mismo *GO:process* participan muchas proteínas. Por eso buscamos en el detector pares *GO:binding-GO:process*, donde *GO:process* describirá la función reconocida de la proteína, en la que participa el *GO:binding*.

Los diferentes pares *GO:binding-GO:process*, son proporcionados como entrada para el módulo de construcción de diccionarios, de forma que para una misma *GO:binding* se detallan los *GO:process* que representarían diferentes funciones de la proteína (ligados al mismo *GO:binding*) y aquellos que representarían realmente la misma función.

Tambien cuando se indica que una función es desconocida, puede haber más de un término, pero la contabilizamos como una sola, ya que pueden pertenecer a una misma función.

Para poder descartar redundantes, se ha procedido al cálculo de la tasa de redundancia de cada función o proceso y se ha realizado un análisis estadístico comparando los resultados con los que muestra *queryGO* en su página web,

para poder verificar que los datos obtenidos eran coherentes.

Compared term	Aspect	Name	PR	S%
GO:0000077	P	DNA damage checkpoint	7,703.91	100.00
GO:0030896	C	checkpoint clamp complex	7,663.29	38.33
GO:0004694	F	eukaryotic translation initiation factor 2alpha kinase activity	6,652.71	11.31
GO:0010998	P	regulation of translational initiation by eIF2 alpha phosphorylation	5,379.66	10.97
GO:0000075	P	cell cycle checkpoint	7,294.33	9.47
GO:0090399	P	replicative senescence	5,480.35	8.16
GO:0016572	P	histone phosphorylation	4,023.83	7.10
GO:0010212	P	response to ionizing radiation	1,851.24	6.75
GO:0031151	F	histone methyltransferase activity (H3-K79 specific)	2,187.22	5.64
GO:0034729	P	histone H3-K79 methylation	2,023.84	5.55

Figura 5: Términos concurrentes de GO:00000077. Imagen obtenida de *quickGO*.

Haciendo referencia a la *figura 5*, nuestro algoritmo obtendría la tasa de concurrencia que sería muy similar a la S% que aparece en la imagen y que es uno de los parámetros de redundancia de *quickGO*[6].

Este tipo de cálculo es el más costoso de todo el proyecto, no por la dificultad de los cálculos, sino por la cantidad de información que se dispone de los genes haciendo que pueda tardar varios días solo en obtener las redundantes de un término.

```
real    7m23.363s
user    0m3.792s
sys     0m2.852s
```

Figura 6: Tiempo en la obtención de redundantes del GO ID GO 0000077 obtenidos con el código antes de la optimización.

Como se puede ver en la *figura 6*, el tiempo en obtener los términos redundantes del GO:0000077 es de 7,4 min(443s), este término tiene 9174 genes.

El GO:0005524 tiene 6956681. Por lo que teniendo en cuenta el tiempo del GO:0000077 el calcularía:

$$\frac{443\ s}{9174\ gen} * 6956681\ gen = 3{,}88\ días \qquad (2)$$

Si tenemos en cuenta que, entre los redundantes de términos y de funciones, la suma de todos ellos es de más de 100.000.000. La obtención de todos los redundantes tardaría alrededor de 2 meses. No obstante, el tiempo de obtención no es lineal ya que, a mayor número de genes a consultar, el tiempo que tardará la base de datos en responder se incrementará exponencialmente.

Por ello, he realizado una segunda variante, donde se realiza la misma operación del cálculo de redundantes, pero mediante consultas y subconsultas más complejas y con ello, toda la carga de ejecución se realizaría en el sistema gestor de bases de datos, SGBD, de *Gene Ontology*.

```
real    0m48.582s
user    0m0.616s
sys     0m0.216s
```

Figura 7: Redundantes GO 0000077 obtenidos con el código optimizado.

Tal y como se observa en la *figura 7*, obtenemos una reducción de tiempo 443/49 = 9 veces más rápido. Al utilizar este segundo método el SGBD, se satura más rápido y comete más fallos. De todas formas, la velocidad de obtención de los redundantes aumentará considerablemente.

Aún así, el tiempo de obtención de redundantes tardará más del doble de tiempo, contando con los fallos que existen al perder la conexión con las bases de datos o incluso a las saturaciones de esta, teniendo que volver a realizar la operación.

He verificado los resultados obtenidos por el módulo de redundantes y los que indican *quickGO*.

GO ID	plot_go_redundant	quick GO
GO:0000077	100%	100%
GO:0030896	38.95%	41%
GO:0000075	7.74 %	9.08 %
GO:0031151	7.02 %	5.85 %
GO:0000076	4.02 %	4.30 %
GO:0010998	5.28 %	5.58 %

Tabla 1: Redundantes de GO:0000077

Como podéis visualizar en la *tabla 1*, los resultados son aproximados, pero tienen una tasa de error de hasta un 2% con los datos comparados. No obstante, *quickGO* no indica la versión de la base de datos que se está utilizando. Yo, en este caso utilicé la base de datos actualizada, ya que me conecté remotamente a un espejo de *Gene Ontology*, donde la base de datos tiene una actualización estable de forma mensual.

Utilizamos como listado de pruebas el fichero *moonsparatest.csv* que contiene un conjunto de proteínas multifuncionales de la especie Homo sapiens con la función canónica, función principal de la proteína, y el resto de funciones descubiertas que están asociadas a la proteína moonlighting. El documento moonsparatest.csv se ha obtenido de la base de datos de proteínas multifunciones, *MultitaskProtDB* [7].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

Tenemos 111 proteínas de entrada, todas son moonlighting, de las cuales detectamos 103 por lo que se obtiene un accuraccy de 93%. Si únicamente tenemos en cuenta la decisión de si es moonlighting o no, aún así no deja de ser del todo fiable debido a que carecemos de proteínas clasificadas como no moonlighting. Por lo que solo podemos trabajar con verdaderos positivos y falsos negativos.

Algunas de las proteínas que indica que no lo son moonlighting son porque las bases de datos de *Gene Ontology* tienen pocos datos.

A la hora de obtener un accuraccy de las funciones clasificadas, procesos con sus funciones asociadas y las funciones extra que pertenecen a la proteína detectada. No sería posible ya que, a la hora de evaluar los aciertos de funciones clasificadas, procesos y las funciones extra que están asociadas a resultados pasan dos cosas:

1: Algunos de los resultados no están registrados, por lo que podría ser un falso positivo (FP), pero también puede ser un verdadero positivo (TP) y que simplemente haya detectado una nueva función moonlighting. Como no es posible obtener estos datos se han realizarán comprobaciones por el método de reconstrucción del programa y comparando con un método más simple y no basado biológicamente a diferencia del funcionamiento del detector.

Las comprobaciones de reconstrucción del programa que utilizo para validar los resultados son:

- Por cada actividad funcional que este en los resultados del detector verificar que este dentro del diccionario de include. Accuracy: 100%.

- Por cada actividad funcional que esté en los datos de entrada del detector y no esté en los resultados verificar que esté dentro del diccionario de include.

  Accuracy: 95%, las que fallan es porque a nivel de bases de datos está puesto como actividades funcionales, pero realmente están más próximas a funciones binding.

- Por cada actividad funcional que esté en los datos de entrada del detector y no esté en los resultados verificará que esté dentro del diccionario de include. Accuracy: 100%.

- Por cada función binding que esté clasificada en un proceso o asignada como extra en resultados verificar que no esté dentro de redundantes de actividad funcional. (gof) y verificaremos que aparezcan en las actividades funcionales de resultados.

  Resultado 98,4%. En este caso no es accuracy, ya que las que no aparecen es porque han sido descartadas por ser redundantes y el porcentaje es solo informativo para saber la cantidad que se descartan.

- Crear un fichero con todas las funciones redundantes que han sido descartadas (binding que aparecen en activity), es decir, que no aparecen en resultados, y su porcentaje de redundancia.

  Con esto podemos ver si existe algún problema al considerar una función redundante, ya que actualmente consideramos redundantes a todas las que tienen más de un 3% de aparición, ya que es un número importante en las distribuciones normales y con la que consideraríamos no redundantes a las que estuvieran en al principio de la campana de Gauss.

- Para las funciones binding de Lextra verificará que aparezcan en algún redundante de proceso. Si aparecen en un fichero gop que el nombre de este fichero no aparezca como datos de entrada go process. Accuracy: 100%.

- Para cada función se verifica si tienen alguna palabra que aparezca en el nombre de un proceso, y si es así, verificar porque no aparecen en los resultados.

2: La otra casuística consiste en que los resultados que sí que aparecen en las clasificaciones realizadas por los científicos, moonsparatest, y que tienen que dar los mismos resultados, están escritos con lenguaje natural.

Tanta es la diferencia que no se puede validar de forma automática y se tienen que evaluar de forma manual. Igualmente, al ser un proyecto de investigación no desarrollado anteriormente no se dispone de ningún listado de proteínas que no sean moonlighting. No obstante, realizando esta comprobación de forma manual los datos obtenidos sobre las funciones son: 82 TP y 125 FP, las cuales no han sido detectadas, ya que no aparecen en las bases de datos.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

Por lo que tendríamos una precisión del 39%, pero si contamos solo los que pueden ser detectados porque estas sí que aparecen en las bases de datos. Tendríamos 31 FP, y la precisión seria de 73% y en este caso aún contamos con algunas limitaciones, ya que el diccionario de redundantes actual es limitado, por lo que a medida que el diccionario aumente, la precisión aumentará

4. Conclusiones

Se ha conseguido la detección de proteínas moonlighting aunque se tendrán que verificar los resultados con un conjunto de test donde hubieran proteínas que no fueran moonlighting, para poder evaluar los resultados teniendo Falsos Negativos y Verdaderos Negativos.

A nivel personal, he podido aprender a trabajar con grandes conjuntos de datos, ya que las bases de datos con las que he trabajado tienen millones de atributos y además se encuentran en lenguaje natural, impidiendo hacer de un modo sencillo los machings.

Por ello, al enfrentarme a problemas de bigdata, el principal foco es hacer pruebas a pequeña escala, ya que las ejecuciones para obtener resultados pueden demorar meses.

Además, hay que tener en cuenta que los resultados puede que no sean los esperados o que existan errores, esto supone un enorme coste a nivel de tiempo y de recursos.

Agradecimientos

Me gustaría agradecer toda la ayuda prestada al tutor del proyecto Mario Huerta, ya que, gracias a sus puntos de vista y a la guía en la resolución de los numerosos problemas surgidos, el resultado ha podido ser el esperado.

También me gustaría agradecer el soporte que me ha brindado mi familia y mis amigos.

Bibliografía

[1] Gene Ontology(data): Base de datos de los genes de Gene Ontology Consortium.

[2] Uniprot(data): Base de datos de proteínas de European Bioinformatics Institute.

[3] Análisis bioinformático de las proteínas multifuncionales (moonlighting): Contiene información sobre las proteínas moonlighting de Sergio Iván Hernández Ranzani.

[4] Do protein-protein interaction databases identify moonlighting proteins?;de Gómez A, Hernández S, Amela I, Piñol J, Cedano J, Querol E.

[5] Goatools 0.6.10: Herramienta para la búsqueda de información de genes de Python software fundation.

[6] QuickGO: Herramienta para la búsqueda de inform ción degenes de European bioinformatics institute

[7] MultitaskProtDB: a database of multitasking proteins; de Hernández S, Ferragut G, Amela I, Perez-Pons J, Piñol J, Mozo-Villarias A, Cedano J, Querol E.

[8] Bioinformatics and Moonlighting Proteins; de Hernández S, Franco L, Calvo A, Ferragut G, Hermoso A, Amela I, Gómez A, Querol E, Cedano J.

[9] Can bioinformatics help in the identification of moonlighting proteins?;de Hernández S, Calvo A, Ferragut G, Franco L, Hermoso A, Amela I, Gómez A, Querol E, Cedano J.

APÉNDICE

1. Diagrama de flujo en fase de toma de decisión:

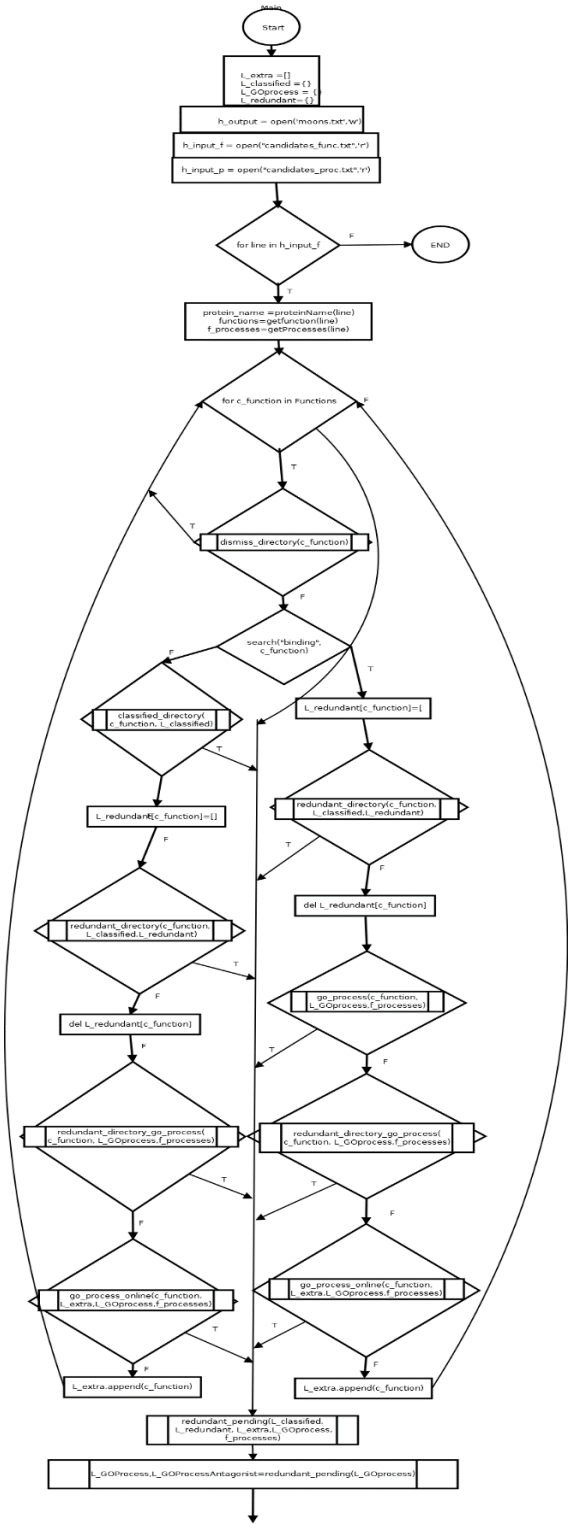

Figura 1: Esquema en detalle del algoritmo de árbol de decisiones de moonDetector en la fase de toma de decisiones.

2. Diagrama de flujo en fase de validación:

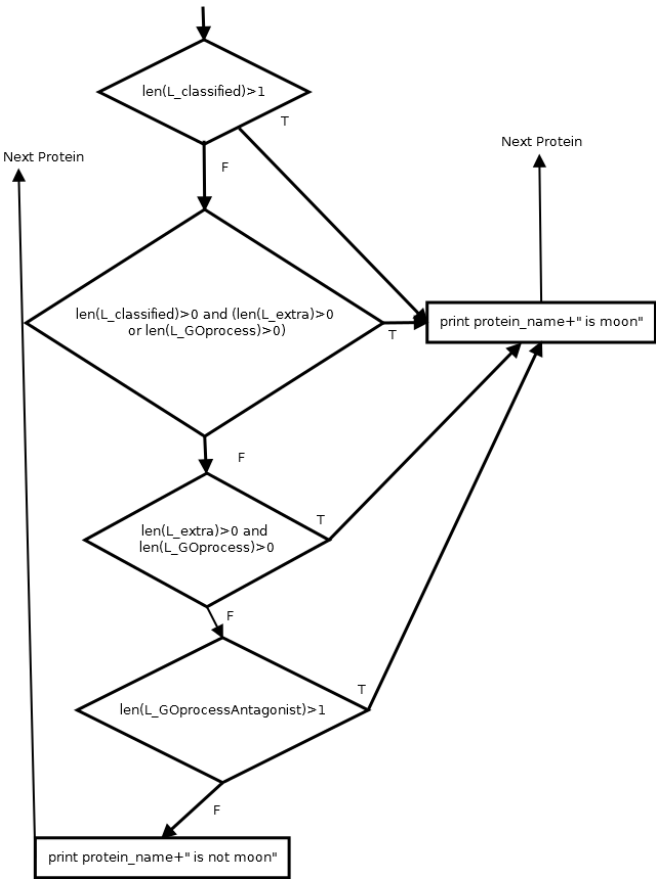

Figura 2: Esquema en detalle del algoritmo de árbol de decisiones de moonDetector en la fase de validación.