

UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

**DISEÑO E IMPLEMENTACIÓN DE NÚCLEO DE SISTEMA
DE DISEÑO DE PARTIDORES PARA REACCIONES EN
CADENA DE POLIMERASA**

**MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO
E INGENIERO CIVIL EN COMPUTACIÓN**

RAUL ESTEBAN ALIAGA DIAZ

**PROFESOR GUÍA:
ALEJANDRO MAASS SEPÚLVEDA**

**MIEMBROS DE LA COMISIÓN:
NANCY HITSCHFELD KAHLER
SERGIO F. OCHOA DELORENZI**

**SANTIAGO DE CHILE
ABRIL 2010**

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO
INGENIERO CIVIL EN COMPUTACIÓN
POR: RAUL ESTEBAN ALIAGA DIAZ
FECHA: 20 DE ABRIL DE 2010
PROF. GUÍA: ALEJANDRO MAASS

DISEÑO E IMPLEMENTACIÓN DE NÚCLEO DE SISTEMA DE DISEÑO DE PARTIDORES PARA REACCIONES EN CADENA DE POLIMERASA

La reacción en cadena de polimerasa (PCR, por sus siglas en inglés) es un proceso ubicuo en laboratorios biológicos. Su uso específico que nos interesa en este trabajo es el de bio identificación de micro organismos en muestras biológicas ambientales, área conocida como *metagenómica*.

Un PCR exitoso requiere pequeñas secuencias de nucleótidos llamadas “partidores”, las cuales deben satisfacer variadas condiciones termodinámicas y biológicas. El proceso de diseño de ellos es un tema de amplio estudio en bioinformática, pues es un proceso maduro y que necesita cada vez más contar con predicciones computacionales de su desempeño.

El objetivo de esta memoria es desarrollar una herramienta de uso intensivo para el diseño de partidores de PCR en metagenómica, basándose en el trabajo previo del Laboratorio de Bioinformática y Matemática del Genoma (LBMG) de la Universidad de Chile.

El sistema existente es una colección de programas, algunos de cómputo paralelo, que permiten definir un universo taxonómico, calcular partidores para dicho universo y simular su comportamiento numéricamente. Sin embargo, el sistema es de difícil operación e insuficiente extensibilidad.

La metodología para este trabajo consiste de una revisión bibliográfica del estado del arte en la literatura científica y del LBMG, para posteriormente proceder a un re diseño que contemple las extensiones de interés y una re implementación del sistema acorde al re diseño.

El resultado obtenido es el núcleo de la herramienta, con un diseño claro y escalable, junto con su correspondiente implementación, que tiene las mismas funcionalidades del sistema existente exceptuando la paralelización. Se obtiene además una especificación de las cualidades que se deben añadir o completar para contar con un sistema finalizado y operativo de uso intensivo.

Se concluye que el trabajo realizado representa un avance importante en la sistematización del diseño de partidores para PCR sobre la cual basar trabajos futuros de extensión, aplicación a distintos dominios biológicos y como muestra de desarrollos concretos realizados íntegramente en el LBMG.

Agradecimientos



Índice general

1. Introducción	1
2. Conceptos Preliminares	4
2.1. Marco biológico	4
2.2. PCR	5
2.3. Diseño de partidores	7
2.4. Principales herramientas de diseño	10
3. Estado del Arte	12
3.1. Trabajo previo en LBMG	12
3.2. Herramientas relacionadas	15
3.2.1. Primers4clades	15
3.2.2. UniPrime2	17
3.2.3. Pythia	18
3.2.4. PrimerHunter	19
3.3. Requerimientos del Software	21
3.4. Comparación de las Herramientas	23
3.5. Estrategia de Desarrollo y Ambiente Operacional	26
3.6. Comparación de sistema anterior y nueva formulación	28

4. Diseño del Sistema	32
4.1. Arquitectura del Sistema	32
4.1.1. Componentes	32
4.1.2. Dinámica del Sistema	34
4.1.3. Lenguaje de desarrollo	35
4.1.4. Patrones de diseño considerados	36
4.2. Diseño detallado y evaluación	36
4.2.1. PrimerConstants	37
4.2.2. ObjectsCollection	38
4.2.3. Preprocess	47
4.2.4. PrimerGenerators	51
4.2.5. PrimerEnhancers	53
4.2.6. DesignStrategies	63
5. Implementación del Sistema	66
5.1. Entorno de Trabajo	66
5.2. Implementación de Clases	67
5.2.1. PrimerConstants	67
5.2.2. ObjectsCollection	67
5.2.3. Preprocess	68
5.2.4. PrimerGenerators	69
5.2.5. PrimerEnhancers	69
5.2.6. DesignStrategies	71
6. Extensiones	73

6.1. Extensiones directas	74
6.1.1. Entorno de ejecución y pulido	74
6.1.2. Paralelización	75
6.1.3. Interfaz web	76
6.2. Extensiones deseables	78
6.2.1. Extensiones adicionales	79
7. Trabajo Futuro y Conclusiones	80
7.1. Trabajo futuro	80
7.2. Conclusiones	80
7.2.1. Observaciones finales	82

Capítulo 1

Introducción

En Biología Molecular, la Reacción en Cadena de Polimerasa (PCR, por sus siglas en inglés) es una técnica para amplificar la cantidad de una copia individual o varias copias de un segmento de ADN en varios ordenes de magnitud. El proceso consta de una serie de etapas y elementos parametrizables y configurables para realizar una amplia gama de manipulaciones genéticas. Actualmente es una técnica común e incluso indispensable en las investigaciones biomédicas y laboratorios de investigación biológica para una variedad de aplicaciones, como la clonación de ADN para secuenciamiento, análisis funcional de genes, diagnóstico de enfermedades infecciosas como AH1N1, identificación de huellas genéticas, entre otras.

Una componente de principal relevancia del proceso son los partidores o *primers*, que son pequeños fragmentos de ADN que definen la secuencia que se ha de amplificar y al momento de diseñar un PCR particular, deben ser escogidos considerando que su rol en la amplificación será lo suficientemente satisfactorio pese a las complejidades biológicas que intervienen. En este sentido, un apoyo bioinformático es clave, pues el proceso de diseño de *primers* para PCR se puede tratar computacionalmente de modo de obtener secuencias que satisfagan las condiciones específicas del proceso de PCR que se espera realizar y además estimar la eficacia de amplificación de un *primer* antes de usarlo.

El Laboratorio de Bioinformática y Matemática del Genoma (LBMG) del CMM, de la Universidad de Chile, ha desarrollado el apoyo bioinformático para realizar PCR en aplicaciones en la industria minera, esperando ampliarlo a otros dominios. El enfoque del LBMG ha ido en la línea de extender el proceso de PCR original para amplificar no sólo una secuencia de ADN, sino que un conjunto de ellas perteneciente a una familia taxonómica de microorganismos cuya presencia se busca detectar en muestras biológicas ambientales, lo que se denomina como **Metagenómica**, para realizar bioidentificación de especies, como por ejemplo de bacterias y archeas acidofílicas ferrosas oxidadoras de hierro, encontradas por primera vez en las minas ácidas de California descubiertas en 1983.

El sistema se desarrolló originalmente el año 2004 y actualmente consiste de una colección de *scripts* paralelizables, que toman un conjunto de secuencias obtenidas de una base de datos, calculan varios parámetros sobre ellas para filtrar los conjuntos de *primers* y finalmente simula la hibridación del PCR con algunas de las secuencias objetivo para quedarse con un conjunto final. Este trabajo fue creciendo de forma paulatina a medida que se realizaba investigación aplicada pionera en el área, hasta que finalmente se contó con un sistema utilizable por las personas que lo desarrollaron que cumple con las demandas actuales.

Por una parte, la aplicación exitosa de la investigación realizada y el conocimiento generado durante ella, motivaron una amplia gama de líneas de investigación, pero por otra parte estas ideas y las operaciones cotidianas se ven limitadas por las dificultades intrínsecas de un sistema que si bien es operativo, cuenta con escasísima documentación, diseño de muy alto nivel y bajo nivel de mantención.

Así, el objetivo de esta memoria es desarrollar una herramienta de diseño de *primers* que satisfaga:

- Tener las mismas funcionalidades de cálculo del sistema existente.
- Que permita ser usada intensivamente por varios usuarios del LBMG.
- Contar con mejoras básicas para la operación del sistema, como guardar los resultados de cálculo en pasos intermedios, una forma más simple de operarlo, entre otras.
- Una implementación basada en un diseño que provea garantías mínimas de extensibilidad y que considere el estado del arte en la literatura científica.

Para lograr el objetivo, la metodología de trabajo consistió en una revisión bibliográfica para conocer el estado del arte de la investigación del área, junto con una revisión y documentación del trabajo existente del LBMG, que posteriormente derivó a un diseño orientado a objetos del software y una implementación iterativa de los distintos módulos diseñados. Finalmente, siempre se mantuvo en consideración que el alcance del trabajo fuera lo suficientemente acotado para ser realizado en el plazo de una memoria, pero lo suficientemente completo para ser autocontenido a la vez.

El contenido de la memoria se organiza como sigue:

- Una sección de **Conceptos preliminares** de biología y relacionados con el proceso PCR en particular, con los conceptos básicos que permitan comprender lo expuesto en el resto de la memoria.
- Revisión del **Estado del arte** en la literatura científica y del LBMG.
- El **Diseño del sistema**.

- La **Implementación del sistema**.
- Una descripción de las **Extensiones** que es posible de realizar.
- Cierre con **Trabajo Futuro y Conclusiones**.

Capítulo 2

Conceptos Preliminares

2.1. Marco biológico

Todo ser vivo está constituido por células; en otras palabras, la materia viva o protoplasma se encuentra organizada en unidades discretas y complejas llamadas células. La célula es la unidad de estructura y de funcionamiento de la materia viviente. Toda célula deriva de otra célula pre existente. Se denomina célula **procariótica** a aquella que carece de núcleo organizado encerrado en una membrana, con su material nuclear flotando libremente en la célula. Las células que posee su núcleo dentro de una membrana se denominan **eucarióticas**.

Pese a estas diferencias, todas las células tienen la propiedad de contener los denominados **cromosomas** en su material nuclear. Estos son corpúsculos, generalmente de forma de filamentos, que controlan el desarrollo genético de los seres vivos. Las células de los individuos de una especie determinada suelen tener un número fijo de cromosomas, que en las plantas y animales superiores se presentan de a pares. En el caso del ser humano, este tiene 23 pares de cromosomas. Las células reproductoras tienen por lo general la mitad de los cromosomas presentes en las corporales o somáticas, de modo que en la fecundación se disponen en el nuevo organismo de a pares con el aporte de las células reproductoras de cada sexo.

Los cromosomas están constituidos por cadenas lineales de ácido desoxirribonucleico (**ADN**) y por proteínas, denominadas histonas, que empaquetan el ADN en unidades de repetición denominadas nucleosomas. Las cadenas de ADN están estructuradas en unidades llamadas **genes**, sintetizadores de proteínas específicas, cada uno de los cuales posee por término medio del orden de 1000 a 2000 pares de nucleótidos. Para producir las proteínas, la información del ADN se transcribe a una molécula intermedia llamada **ARN mensajero** (mRNA), parte del ADN se desenrolla de su empaquetamiento cromosómico y las dos cadenas se separan en una porción de su longitud, actuando una de ellas como plantilla sobre la que se forma el ARN mensajero (con la ayuda de una enzima denominada ARN polimerasa)

que da las instrucciones para ensamblar los aminoácidos en el orden preciso para formar la proteína.

El ADN está formado por un azúcar (2-desoxi-D-ribosa), ácido fosfórico y bases nitrogenadas: adenina, guanina, citosina y timina. Su estructura es la de una doble hélice en la que las bases se encuentran situadas en el interior de la molécula y los grupos fosfato se disponen en el exterior. Las bases nitrogenadas se unen siempre del mismo modo: adenina con timina y guanina con citosina, a través de puentes de hidrógeno (la adenina frente a la timina forma un doble puente de hidrógeno mientras que la guanina frente a la citosina forma un enlace triple de hidrógeno) y dicha propiedad es llamada **complementariedad**. Esta estructura se mantiene estable gracias al apilamiento de las bases en el centro de la molécula y las hebras, de orientaciones opuestas, pueden separarse por la acción del calor o de determinadas sustancias químicas. El ADN es portador de la información genética que está codificada en la secuencia de bases (el **genoma**), dependiendo esta del orden en que se presentan. Tiene la capacidad de replicarse y de mutar.

En un **gen**, la secuencia de los nucleótidos a lo largo de la cadena de ADN define una proteína que un organismo es capaz de sintetizar o “expresar” en uno o varios momentos de su vida, usando la información de dicha secuencia. La relación entre la secuencia de nucleótidos y la secuencia de aminoácidos de la proteína es determinada por un mecanismo celular de traducción conocido como código genético. Las proteínas formadas son de importancia central pues son las que determinan las reacciones químicas que implementan la diversidad de mecanismos celulares.

2.2. PCR

Este proceso es usado para amplificar la cantidad presente en muestras biológicas de regiones específicas de hebras de ADN (el ADN objetivo). Esta región puede ser un gen, parte de un gen, o una secuencia no codificadora¹. Estos métodos típicamente amplifican fragmentos de ADN en una medida de 1 kbp², aunque algunos métodos pueden amplificar hasta 4 kbp. Estos fragmentos se denominan **amplicones**.

Un montaje básico de PCR requiere varias componentes[26]:

- Una plantilla de ADN (o **template**) que contiene la región de ADN objetivo que será amplificada, de un tamaño entre 2 a 10 kbp dependiendo de la aplicación puntual.
- Dos *primers* que son complementarios al extremo de la hebra de ADN llamada 3', en sentido positivo y negativo de recorrer dicha hebra. Su tamaño está típicamente entre los 10 a 30 pares de bases.

¹Secuencias que no se traducen a proteínas.

²Donde kbp son “*kilo base pairs*”, es decir, 1000 pares de bases nitrogenadas de la cadena de ADN.

- **Polimerasa Taq** u otra ADN polimerasa que pueda utilizarse a una temperatura cerca de 70 °C.
- **Nucleósidos** o dNTP's, los bloques a partir de los cuales la polimerasa sintetiza una nueva hebra de ADN, extendiendo la hibridación de los *primers*.
- Una **Solución de Buffer**, que consiste en una solución química que provee una estabilización del pH en el medio donde se llevará a cabo la amplificación.
- **Cationes divalentes**: Iones de magnesio o manganeso, que intervienen en la fisicoquímica del proceso.
- **Cationes monovalentes**: Iones de potasio o sodio, también importantes para la fisicoquímica.

En general el PCR se lleva a cabo en un volumen de reacción de 10 a 200 μl en pequeños tubos (0,2 – 0,5 ml) dentro de un termociclador, instrumento que calienta y enfría los tubos para lograr las temperaturas requeridas en cada paso.

El procedimiento consiste de una serie de 20 a 40 cambios de temperatura llamados **ciclos**. Cada ciclo consiste de 2 a 3 cambios discretos de temperatura, cuya duración y valores específicos dependen de varios parámetros, tales como la enzima (polimerasa) utilizada, la concentración de iones divalentes y nucleósidos presentes en la reacción, y la temperatura de *melting*³ de los *primers*.

Los pasos son[26]:

1. **Inicialización**: Este paso consiste en calentar la reacción a una temperatura de 94 – 96 °C, la cual se mantiene por 1 – 9 minutos. Este paso sólo es requerido por polimerasas que se activan por calor.
2. **Desnaturalización**: Es el primer evento que participa de un ciclo y consiste en calentar la reacción a 94 – 98 °C por 20 – 30 segundos. Esto provoca la separación del *template* y los *primers* al romper los puentes de hidrógeno entre bases complementarias de hebras de ADN y las convierte en cadenas de bases.
3. **Alineamiento**: La temperatura se disminuye a 50 – 65 °C por 20 – 40 segundos, permitiendo que los *primers* se unan con las cadenas de bases para armar un trozo de hebra. En esta etapa es muy importante que el *primer* sea un buen matching de la cadena de *template*, pues a partir de la porción de hebra que ambos forman se extiende la hebra completa que se amplificará.

³Temperatura a la cual una doble hebra de ADN se separa, rompiendo sus puentes de hidrógeno.

4. **Elongación:** En este paso la polimerasa sintetiza una nueva hebra de ADN complementaria a la cadena *template* añadiendo dNTP's (nucleósidos) y extendiendo la hebra formada entre los *primers* y el *template*. El tiempo que esto toma depende de la polimerasa y el largo del fragmento que se busca amplificar. En general, en esta etapa se habrá duplicado el número de hebras requeridas. Al final de este paso se vuelve a la etapa de desnaturalización, repitiendo éste y los dos pasos previos, por la cantidad de ciclos que se realizarán.
5. **Elongación final:** Este paso final se realiza ocasionalmente, a una temperatura de $70 - 74^\circ\text{C}$ por $5 - 15$ minutos después del último ciclo para asegurar que cualquier cadena que esté parcialmente unida sea extendida completamente.
6. **Conservación:** Este paso se ejecuta a $4 - 15^\circ\text{C}$ por un tiempo suficiente para conservar la reacción por un corto periodo de tiempo.

Para corroborar que el PCR generó el fragmento de ADN esperado, se usa una técnica llamada “Electroforesis en gel de agarosa”, que permite situar en niveles de medición la cantidad de material genético que exista en la muestra (ver Figura 2.2).

2.3. Diseño de partidores

El objetivo es obtener un balance entre dos objetivos: especificidad y eficiencia de la amplificación[6]. Especificidad puede definirse como la no hibridación de un *primer* a zonas del ADN que no sean las deseadas. Eficiencia se define como cuán cerca una pareja de *primers* amplifica a una tasa del doble de producto por cada ciclo del PCR. Diversos criterios y restricciones[6] son usados para proveer *primers* específicamente para una aplicación:

1. **Largo del *primer*:** Puesto que la especificidad del *primer* y las temperaturas de alineamiento (*annealing*) y desnaturalización (*melting*) son dependientes parcialmente del largo del *primer*, este parámetro es crítico para un PCR exitoso.
2. **Contenido GC⁴:** Una característica importante del ADN, provee información acerca de la fuerza del alineamiento.
3. **Estabilidad y Especificidad de los extremos:** Los extremos de las hebras de ADN pueden orientarse, siendo un extremo llamado $5'$ y el otro $3'$, y éstos deben ser estables y específicos, respectivamente.
4. **Temperatura de desnaturalización y de alineamiento (T_m y T_a):** Estos parámetros son dependientes el uno del otro, y dependen también de la cantidad de contenido GC y el largo del *primer*.

⁴Cantidad de nucleótidos G y C respecto del total.

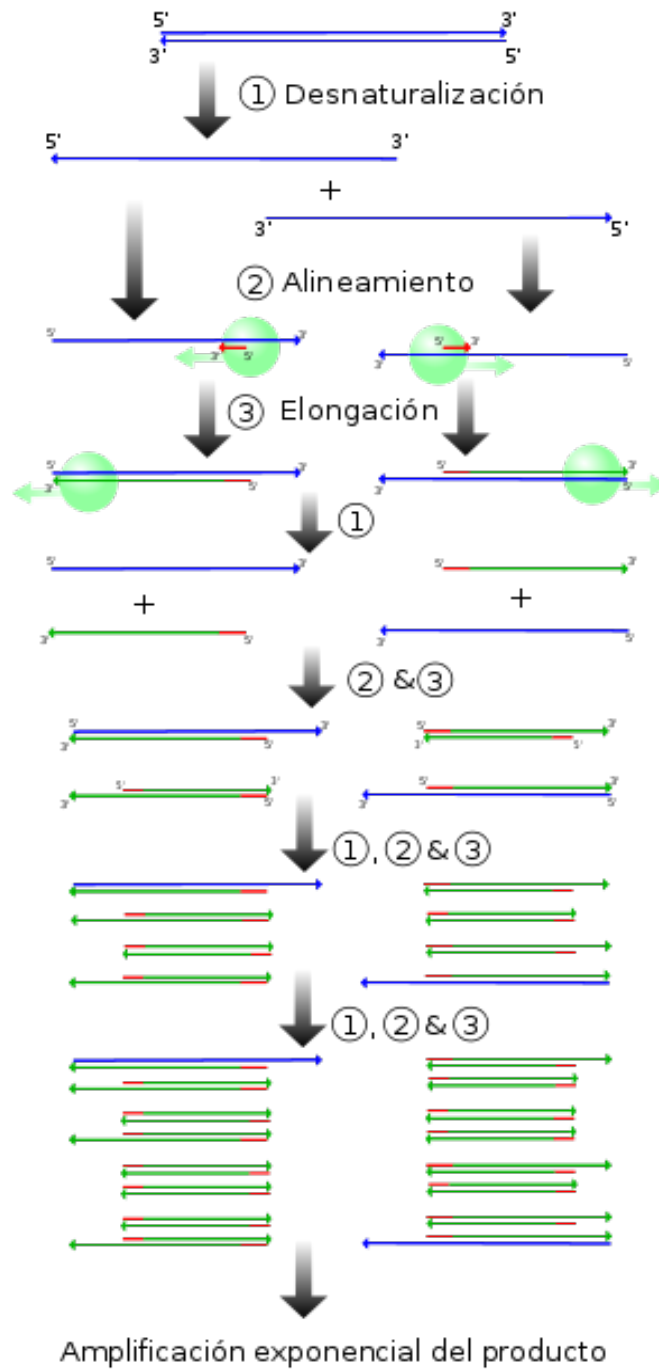


Figura 2.1: Esquema Gráfico del Proceso PCR

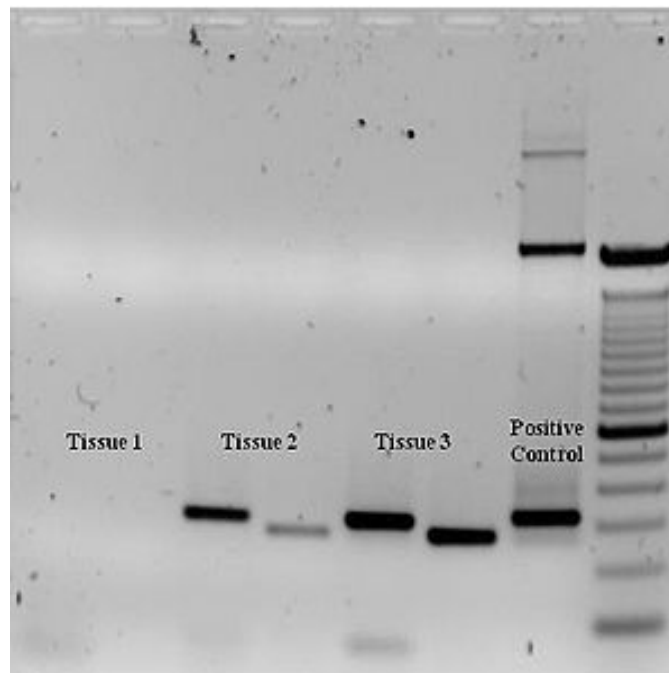


Figura 2.2: Productos de PCR después de aplicar la técnica de Electroforesis en gel de agarosa. Dos conjuntos de *primers* fueron usados para amplificar una secuencia objetivo de tres muestras distintas. No hay amplificación en la muestra número 1; las bandas de ADN en las muestras 2 y 3 indican una amplificación exitosa de la secuencia objetivo. El gel también muestra un control, una escala que contiene fragmentos de ADN de tamaño conocido para medir las bandas de los PCR's experimentales.

5. **Estructuras secundarias:** Así se llaman debido a que la hibridización ocurre de forma espontánea entre las hebras presentes, es necesario considerar que los *primers* no hibridicen entre sí mismos, entre las parejas de *primers*, que no formen “*loops*”, ni que tampoco los *primers* hibridicen fuera de las zonas que uno desea para generar los amplicones.

Dependiendo de las variantes de PCR para las cuales se necesite diseñar *primers*, los criterios pueden aumentar y ser más específicos. Algunas variantes son:

1. **Multiplex PCR:** Utiliza varias parejas de *primers* para hibridizar con varias secuencias objetivo y obtener varios amplicones en un mismo PCR.
2. **Asymmetric PCR:** Preferentemente amplifica una hebra del ADN objetivo sobre la complementaria, vía proveer un exceso de *primers*, entre otras consideraciones termodinámicas.
3. **Nested PCR:** Utilizada para aumentar la especificidad de la amplificación. Dos conjuntos de *primers* son utilizados en reacciones sucesivas. En el primer PCR se amplifican ciertas regiones de ADN las cuales se ocupan como sustrato para un segundo PCR, y así obtener mejor especificidad.
4. **Quantitative PCR:** Es usado para medicar la cantidad específica del ADN de una muestra. Ocupa métodos de fluorescencia con fluoróforos intercalados en el ADN o bien oligonucleótidos modificados para cuantificar el producto que se va formando a lo largo de los ciclos del PCR.
5. **Touchdown PCR:** La temperatura de alineamiento es gradualmente disminuida a lo largo de los ciclos. Esto es para mejorar la especificidad del alineamiento, mientras que la menor temperatura al final mejora la amplificación.

2.4. Principales herramientas de diseño

Debido a que muchos de los principales criterios de diseño son cálculos automatizables, existen diversas herramientas bioinformáticas que permiten asistir el trabajo de algunas partes del diseño de *primers* para varias aplicaciones y variantes de PCR, entre ellas podemos citar de forma ilustrativa a[1] (ver Cuadro 2.1) :

Cuadro 2.1: Herramientas para diseño de *primers* ampliamente conocidas.

CODEHOP	CO nsensus DE generate Hy brid O ligonucleotide P rimers; diseño de <i>primers</i> degenerados a partir de alineamientos múltiples de proteínas.	http://blocks.fhcrc.org/codehop.html
Primer3Plus	La herramienta más popular para diseño de <i>primers</i> .	http://www.bioinformatics.nl/cgi-bin/primer3plus/primer3plus.cgi
The Primer Generator	Herramienta que analiza la secuencia nucleótida original y la secuencia del aminoácido deseado y diseña un <i>primer</i> que contempla criterios a partir de dichas secuencias.	http://www.med.jhu.edu/medcenter/primer/primer.cgi

Capítulo 3

Estado del Arte

3.1. Trabajo previo en LBMG

En el LBMG, se ha trabajado en tratar computacionalmente el proceso de diseño de *primers* para PCR, extendiéndolo a un contexto de **metagenómica**. Es decir, muestras con material genético recuperadas directamente de contextos ambientales. Este enfoque está inspirado en la aplicación directa de bioidentificación de especies presentes en muestras.

Originalmente, el sistema fue desarrollado especialmente para la bioidentificación de microorganismos “biomineros”, es decir, bacterias o archaeas que son relevantes para el proceso de biolixiviación¹. En este contexto, consideraciones biológicas particulares aplicadas al problema permitieron desarrollarlo en el siguiente formato (ver Figura 3.1):

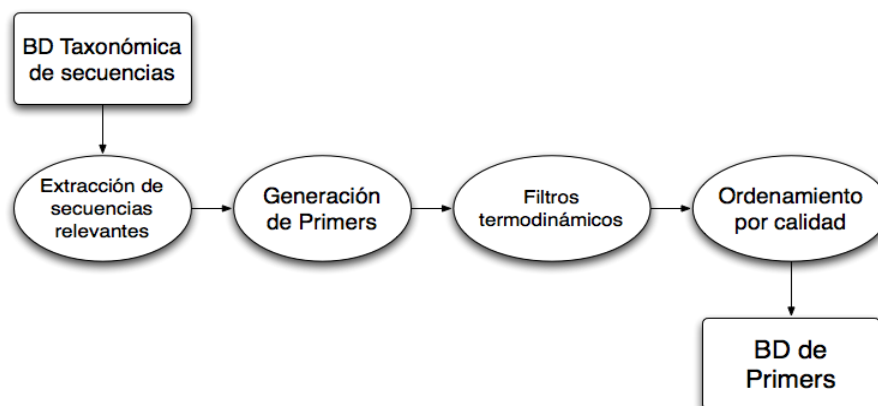


Figura 3.1: Esquema gráfico del proceso bioinformático desarrollado en LBMG.

¹La extracción de metales específicos a partir de rocas minerales mediante el uso de bacterias.

- **Extracción de secuencias relevantes:** De acuerdo a los criterios biológicos adecuados, se extraen las secuencias de la base de datos de la NCBI², para luego conformar una base de datos local enriquecida con datos adicionales a partir de la cual se crearán los *primers*.
- **Generación de *primers*:** Basándose en los trabajos de Miura[19] y SantaLucia[27], que consideran regiones relevantes de hibridación de los *primers* y la termodinámica de la hibridación.
- **Filtros termodinámicos:** Se calculan varios indicadores termodinámicos a partir de la secuencia de cada *primer* que permiten filtrar el conjunto. Estos indicadores buscan medir la factibilidad de que se formen *estructuras secundarias*, es decir, hibridaciones de un *primer* consigo mismo, con una copia de sí mismo, o con su pareja, entre otros.
- **Ordenamiento por calidad:** Entre el *primer* y la secuencia objetivo, se pueden calcular medidas que permiten evaluar la eficacia con la cual se produce la hibridación, y consecuentemente la amplificación deseada. Finalmente, los resultados se muestran en un sitio web (ver Figura 3.2).

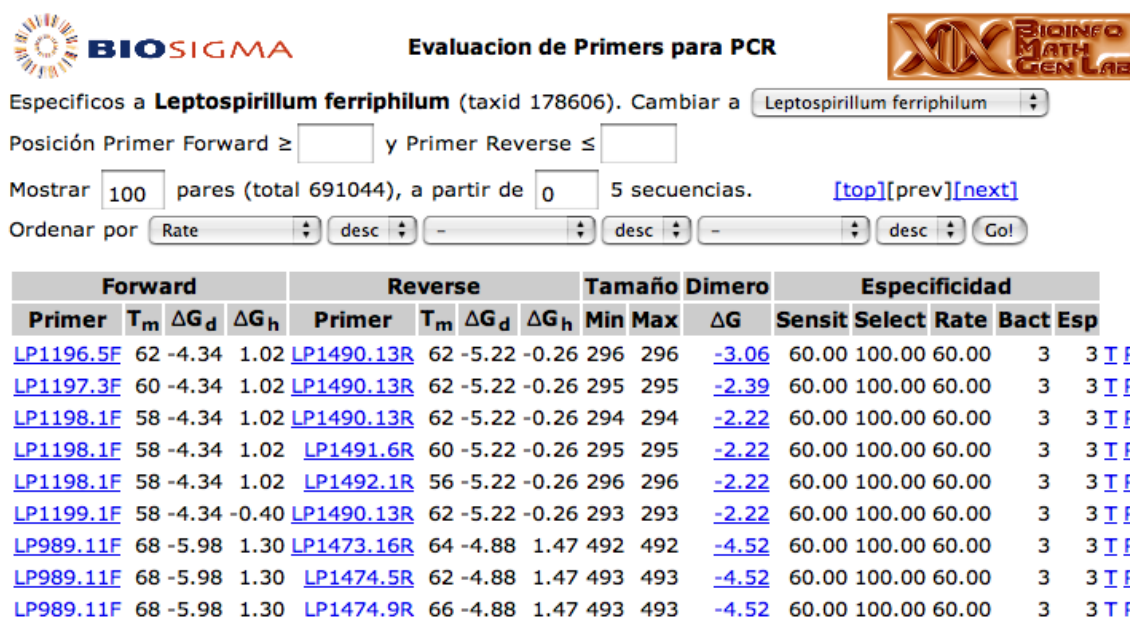
El sistema actual presenta bastantes dificultades que impiden utilizarlo para las diversas necesidades que debería satisfacer. Las principales dificultades son:

1. **Bajo nivel de actualización de secuencias:** Debido a que los datos taxonómicos y las secuencias utilizadas se obtienen de fuentes de datos externas que están en permanente actualización, es necesario a su vez actualizar constantemente las bases de datos de diseño locales. No obstante, este proceso se realiza actualmente de forma manual y con poca periodicidad.
2. **Escasa flexibilidad y mantenibilidad:** El proceso está implementado a nivel de scripts en el lenguaje Perl y *makefiles*, con poca documentación de su funcionamiento y escasa capacidad de mantención o extensión, sin una arquitectura o diseño de alto nivel que permita adaptarlo a nuevos contextos o ejecutarlo de manera intensiva.
3. **Difícil evaluación de resultados:** Lo obtenido del proceso se muestra en un sitio web que se espera sea utilizado por personas del área de biología, y actualmente es complejo evaluar los resultados debido a que son varios parámetros a considerar y no directamente interpretables biológicamente.
4. **Gran número de resultados:** En cuanto al proceso mismo, que esencialmente impone condiciones necesarias sobre los *primers*, el número de posibles pares de candidatos con el modelo actual es muy alto comparado con las posibilidades de aplicar los resultados en un laboratorio biológico. Por ello se busca crear e incluir a este proceso un criterio adicional para evaluar los *primers*, actualmente en desarrollo en el LBMG.

²National Center for Biotechnology Information, institución que alberga numerosas bases de datos genómicas, entre otros recursos.

Algunas de las extensiones que se esperan realizar con el sistema son:

1. **Paralelización de cálculos:** Muchos de los cálculos intensivos que se hacen sobre las secuencias pueden realizarse de manera independiente en distintos computadores, disminuyendo el conjunto de datos con el cual trabajar en ciertas partes del proceso.
2. **Consultas distribuidas:** Del mismo modo que los cálculos, las consultas pueden segmentarse en varias consultas que se ejecutan en distintos computadores.
3. **Mejoras de interfaz web:** La visualización y evaluación de resultados es realizada por expertos de formación biológica interesados en *primers* específicos para sus propias investigaciones y/o aplicaciones, y necesitan por tanto acceder de forma expedita al sistema para usarlo, visualizar e interpretar los resultados, mejorando lo que existe actualmente (ver Figura 3.2).
4. **Benchmark con otros sistemas:** Con el objetivo de capitalizar el proceso en un trabajo científico publicable, es necesario compararlo cualitativa y cuantitativamente con otros sistemas o *pipelines* que realizan tareas similares, y por tanto los requerimientos y restricciones serán similares al trabajo previo científico existente el cual se debe definir e investigar.
5. **Nuevas simulaciones de hibridación:** Este aspecto es de vital importancia y de continua investigación y avance en la comunidad científica, por tanto es necesario que el proceso permita mejorar este aspecto flexiblemente. Es además, el filtro adicional que se espera incluir de los nuevos desarrollos del LBMG.
6. **Actualización automática de bases de datos de secuencias:** Con el fin de evitar políticas de actualización poco frecuentes e inconsistentes, las cuales podrían tener gran impacto en los resultados obtenibles por el proceso, es que se desea contar con esta funcionalidad.
7. **Mejoras algorítmicas:** Varias partes del *pipeline* realizan cálculos que son susceptibles de varias mejoras, ya sea conceptuales o algorítmicas, o de uso de técnicas del área de *High Performance Computing*(HPC).
8. **Nuevas variantes de PCR:** Existen variantes del proceso de PCR para las cuales el sistema podría requerir prontamente extenderse, e implican cambios no sólo paramétricos sino que eventualmente estructurales a varias partes del *pipeline*, y es deseable contar con factibilidad de modificar el sistema en este sentido.



Específicos a **Leptospirillum ferriphilum** (taxid 178606). Cambiar a

Posición Primer Forward \geq y Primer Reverse \leq

Mostrar pares (total 691044), a partir de 5 secuencias. [\[top\]](#)[\[prev\]](#)[\[next\]](#)

Ordenar por

Forward				Reverse				Tamaño Dimero			Especificidad				
Primer	T _m	ΔG_d	ΔG_h	Primer	T _m	ΔG_d	ΔG_h	Min	Max	ΔG	Sensit	Select	Rate	Bact	Esp
LP1196.5F	62	-4.34	1.02	LP1490.13R	62	-5.22	-0.26	296	296	-3.06	60.00	100.00	60.00	3	3 I P
LP1197.3F	60	-4.34	1.02	LP1490.13R	62	-5.22	-0.26	295	295	-2.39	60.00	100.00	60.00	3	3 I P
LP1198.1F	58	-4.34	1.02	LP1490.13R	62	-5.22	-0.26	294	294	-2.22	60.00	100.00	60.00	3	3 I P
LP1198.1F	58	-4.34	1.02	LP1491.6R	60	-5.22	-0.26	295	295	-2.22	60.00	100.00	60.00	3	3 I P
LP1198.1F	58	-4.34	1.02	LP1492.1R	56	-5.22	-0.26	296	296	-2.22	60.00	100.00	60.00	3	3 I P
LP1199.1F	58	-4.34	-0.40	LP1490.13R	62	-5.22	-0.26	293	293	-2.22	60.00	100.00	60.00	3	3 I P
LP989.11F	68	-5.98	1.30	LP1473.16R	64	-4.88	1.47	492	492	-4.52	60.00	100.00	60.00	3	3 I P
LP989.11F	68	-5.98	1.30	LP1474.5R	62	-4.88	1.47	493	493	-4.52	60.00	100.00	60.00	3	3 I P
LP989.11F	68	-5.98	1.30	LP1474.9R	66	-4.88	1.47	493	493	-4.52	60.00	100.00	60.00	3	3 I P

Figura 3.2: Página web mostrando algunos *primers* para identificación de *Leptospirillum ferriphilum*.

3.2. Herramientas relacionadas

Para abordar el desafío de encaminar este trabajo en la dirección correcta, hay que primero determinarla y por tanto es necesario realizar una profunda y detallada revisión bibliográfica, la cual en el área de bioinformática se caracteriza por ser siempre muy abundante y reciente. De lo investigado, se presentan las herramientas que son más relevantes, considerándolas como tales bajo el criterio de cubrir alguna de las características críticas del trabajo desarrollado en el LBMG: Aplicación a bioidentificación y/o filogenética, mejoras en los modelos termodinámicos y definición explícita de secuencias “no objetivo”. A continuación una revisión detallada de dichas herramientas a partir de las publicaciones científicas que las describen. Todas las publicaciones son del año 2008 o 2009.

3.2.1. Primers4clades

Primers4clades[5] es un servicio web que usa árboles filogenéticos³ para diseñar *primers* para PCR con cualidades de especificidad respecto a linaje para estudios de diversidad y metagenómica. Las principales cualidades son la posibilidad de especificar *taxones*⁴ mediante una interfaz interactiva para diseñar *primers* para metagenómica y diversidad. Según los

³Filogenética es el estudio de relación evolutiva entre varios grupos de organismos (especies, poblaciones), el cual es realizado a través de datos de secuenciamiento molecular y matrices de datos morfológicos.

⁴Plural de Taxón (*taxa* en inglés) significa un grupo de uno o más organismos.

autores, es “la única herramienta disponible en la red libremente accesible que usa árboles filogenéticos para interactivamente dirigir la búsqueda de *primers*” [5].

La estrategia de diseño es vía CODEHOP[22] (*CO*nsensus-*DE*generate *Hybrid Oligonucleotide Primer*), y utiliza propiedades termodinámicas para evaluar los *primers* y priorizarlos. Además, realiza una predicción del contenido de información filogenética del amplicón⁵ y visualización de la cobertura de este último.

El software está disponible en <http://maya.ccg.unam.mx/primers4clades/> y <http://floresta.eead.csic.es/primers4clades/>, incluye conjuntos de datos de demo y una detallada documentación. Está escrito usando la librería Bioperl[29], y se contrasta en su uso contra Primaclade[11], Greene SCPrimer[12], PriFi[10], GeneFisher-P[15], QPRIMER web server[14] y Multiplex Server[21].

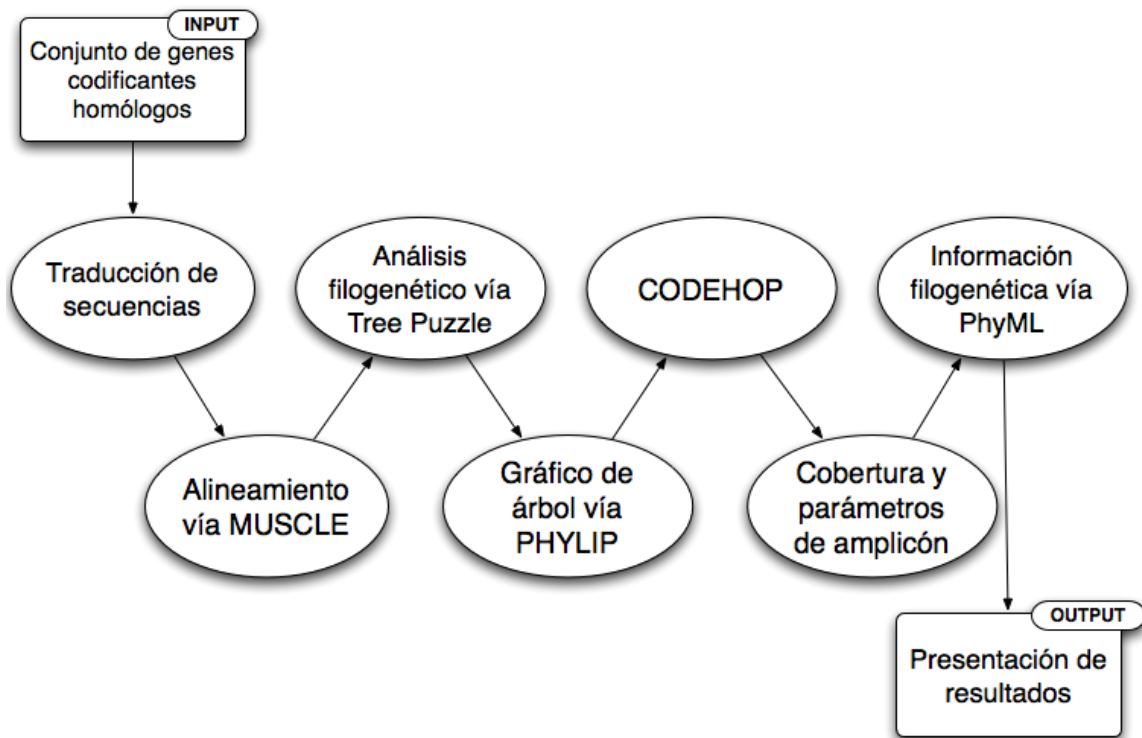


Figura 3.3: Esquema del *pipeline* de Primers4clades.

El *pipeline* (ver Figura 3.3) recibe como entrada un conjunto de genes codificantes homólogos⁶ en formato FASTA⁷, y luego estas secuencias se traducen a proteínas usando tablas de traducción provistas por el usuario, para luego alinearlas usando MUSCLE[8]. Posteriormente se realiza un análisis de máxima verosimilitud filogenética usando Tree-Puzzle[28], una visualización de árbol vía PHYLIP[9] y una extensión a la estrategia CODEHOP[22]. Finalmente

⁵Porción de ADN cuya cantidad es aumentada exponencialmente por el PCR.

⁶Genes cuyas secuencias son similares en un sentido dependiente de lo que se busca observar en la similitud o diferencia.

⁷Formato estándar de secuencias nucleótidas.

se realiza una visualización de la cobertura del amplicón junto a una serie de cálculos termodinámicos utilizando `Amplicon`[13] y un cálculo de la información filogenética con `PhyML`[2].

Las aplicaciones para las cuales fue concebido son estudios de metagenómica enfocados a secuencias y genes que codifican proteínas, para estudiar patrones de expresión de “genes funcionales” en el ambiente, filogenética molecular y epidemiología molecular. La herramienta se validó en varios de sus parámetros con 983 familias de genes ortólogos⁸ compartidas por 19 genomas completamente secuenciados de bacterias rizobiales, con un mayor énfasis en ciertos parámetros más relevantes que otros. Experimentalmente, se contrastó con PCR realizado a fragmentos de secuencias `rpoB`⁹, extraídas de mycobacterias ambientales.

La relevancia de revisar las *features* de esta herramienta es debido a su enfoque filogenético interactivo para diseñar *primers*, además de ser un *pipeline* construido a partir de herramientas previas.

3.2.2. UniPrime2

`UniPrime`[3] es una herramienta para diseño de *primers* universales para cualquier *locus*¹⁰ dado. Sus *features* incluyen: maximización del número de posiciones variables en el fragmento amplificado con un umbral de similaridad provisto por el usuario, evaluación de potencial “*no primer-ización*” de un *primer* (baja estabilidad del extremo 3’), un método y software robustos a condiciones de diversidad genética mientras haya consenso en las secuencias, una medida de “divergencia evolucionaria” y análisis filogenético.

El software es open source (con licencia GPL) y está disponible en <http://habanero.ucd.ie/uniprime2/>, se conecta con el sitio de GenBank¹¹ y automatiza todos los pasos separándolos y proveyendo la posibilidad de correr sólo parte de ellos. Utiliza la librería `Bioperl`[29] (versión 1,4), `Tcoffe` 3.2[20], `PostgreSQL` 7, `Primer3`[24], `PHP` 4,3. La validación experimental incluyó la utilización de `DnaSP` 4,10,9[23] y `PAUP` 2.0b10[30].

El proceso (ver Figura 3.4) consiste en: recibir como input una secuencia de referencia o número de acceso en GenBank, para posteriormente realizar un `blastn`¹² -con algunos parámetros para éste provistos por el usuario- obteniendo así un conjunto de genes ortólogos (parálogos¹³ deben ser removidos por el usuario). Con esto se invoca a `T-Coffee` para estimar el consenso y este resultado se utiliza como input para `Primer3`, y los resultados se filtran

⁸Genes homólogos, i.e. similares, pero que se diferencian vía algún proceso de especiación. Estas secuencias son muy utilizadas en estudios filogenéticos.

⁹Gen bacterial que codifica parte de una enzima que sintetiza ARN.

¹⁰Posición de un gen u otra secuencia significativa en un cromosoma.

¹¹<http://www.ncbi.nlm.nih.gov/Genbank/>

¹²Este programa, dada una secuencia de ADN, retorna las secuencias más similares desde las bases de datos especificadas por el usuario.

¹³Dos genes son **parálogos** cuando son iguales pero están en distintas posiciones del genoma.

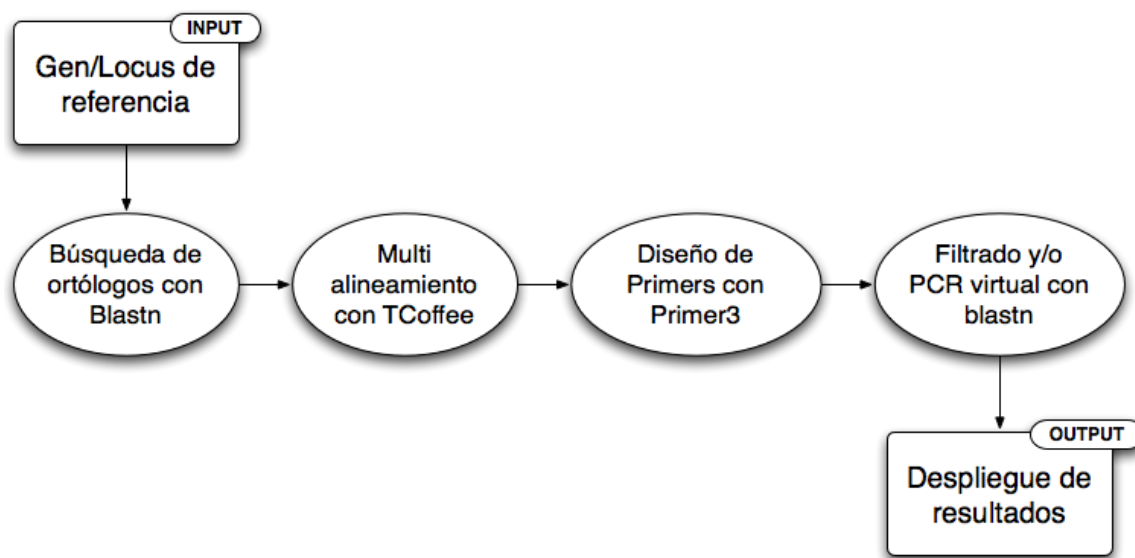


Figura 3.4: Esquema del *pipeline* de UniPrime.

y ordenan para ser expuestos al usuario. Opcionalmente se puede realizar un “PCR virtual” alineando los amplicones con la base de datos de GenBank, mediante `blast` nuevamente.

La aplicación de esta herramienta es a la búsqueda de *primers* universales, así llamados por ser usados en estudios filogenéticos, evolución genética a través de especies, entre otros. Se validó experimentalmente en distintos mamíferos contrastándolo con pruebas de laboratorio.

Posteriormente en UniPrime2[4] se proveyó la opción de realizar el alineamiento con GramAlign[25], delimitar todos los pasos filogenéticamente, correrlo en un servidor web, soportar formato FASTA en el input y mejoras en la visualización de resultados. Se realizó la misma validación experimental que con la primera versión.

Esta herramienta es importante revisar debido a varias ventajas computacionales que incluye, además de ser un *pipeline* e incorporar filogenética.

3.2.3. Pythia

Pythia[17] es la integración de varias herramientas teóricas actuales relacionadas con afinidad de ligamiento de ADN, análisis de equilibrio de reacciones químicas como medida de eficiencia en PCR y evaluación de especificidad. El objetivo es el diseño de *primers* en general, pero con un enfoque termodinámico actual y más significativo. El software está disponible en <http://pythia.sourceforge.net>, y fue concebido para genes con secuencias repetidas (mamíferos).

El proceso (ver Figura 3.5) consiste en: recibir una secuencia, las coordenadas de su locus y

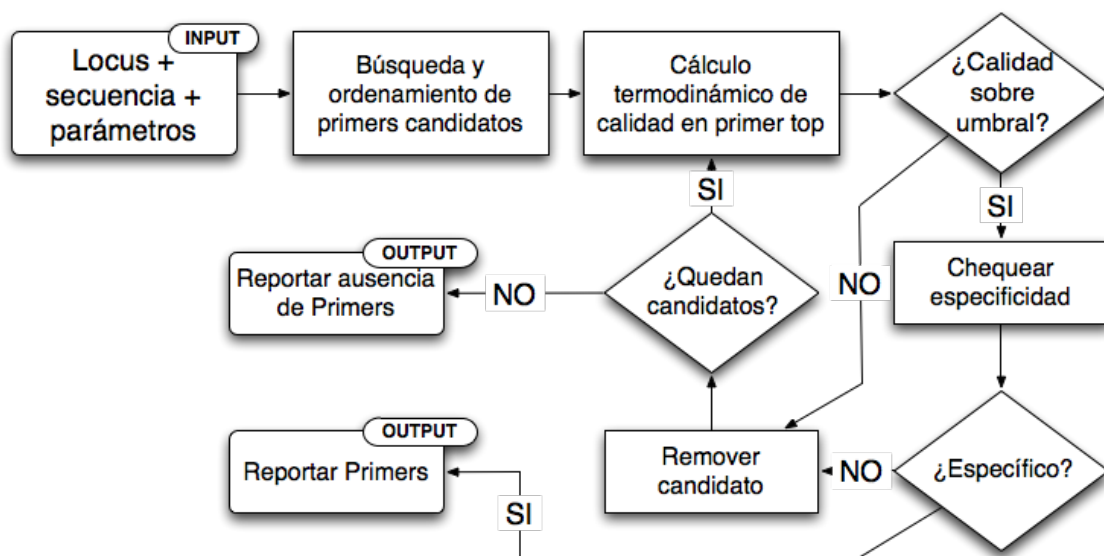


Figura 3.5: Esquema del proceso de Pythia.

parámetros provistos por el usuario. Con estos datos se calculan parámetros termodinámicos básicos de los *primers* y se ordenan según ellos. En el siguiente paso, se calcula la métrica de calidad termodinámica del primer para el mejor candidato, y si ésta está sobre el umbral, se realiza un chequeo de especificidad. De pasar este chequeo, el *primer* es reportado al usuario. De no satisfacerse alguno de los dos últimos chequeos mencionados, el candidato se remueve y se retorna al listado. El cálculo de calidad termodinámica evalúa la factibilidad del *primer* primero usando una *Support Vector Machine* (SVM), y luego un análisis de equilibrio químico completo para posteriormente calcular la métrica de calidad. La especificidad se calcula siguiendo el trabajo de Miura[19] para la estabilidad termodinámica de secuencias.

La validación se llevó a cabo comparando el software con *Primer3* y pruebas de laboratorio, y las aplicaciones esperables son cualquier contexto en que se requiera diseño de *primers* para PCR. La gran ventaja de *Pythia* sobre otras herramientas es que el filtrado de parejas candidatas, la calidad de amplificación y especificidad de dicha amplificación son todos calculados a partir de parámetros termodinámicos de la reacción de PCR, lo cual hace mucho más simple e interpretable la manipulación de parámetros para obtener *primers*.

La incorporación del trabajo de Miura[19] a las mejoras termodinámicas, entre otras consideradas en la herramienta, la hacen interesante para posibles comparaciones de performance o incorporación de sus propuestas de mejoras termodinámicas al *pipeline* de LBMG.

3.2.4. PrimerHunter

PrimerHunter[7] es un software para diseñar *primers* para identificación de subtipos de virus. Recibe como input secuencias objetivo que se desean amplificar y secuencias “no

objetivo” que se desea evitar amplificar. Utiliza el modelo de Santa Lucia[27] para el cálculo de la temperatura de *melting* optimizado mediante el enfoque de programación fraccional para el cálculo de temperaturas de *melting*[16].

El espacio de diseño de *primers* es ligeramente más amplio que el tradicional, al considerar todos los substrings posibles de las secuencias objetivo y permitiendo hibridaciones no perfectas según se especifique en los inputs que entregue el usuario. Considera también mejoras y extensiones posibles de tener específicamente en el ámbito de diseño de *primers* para identificación de subtipos de virus.

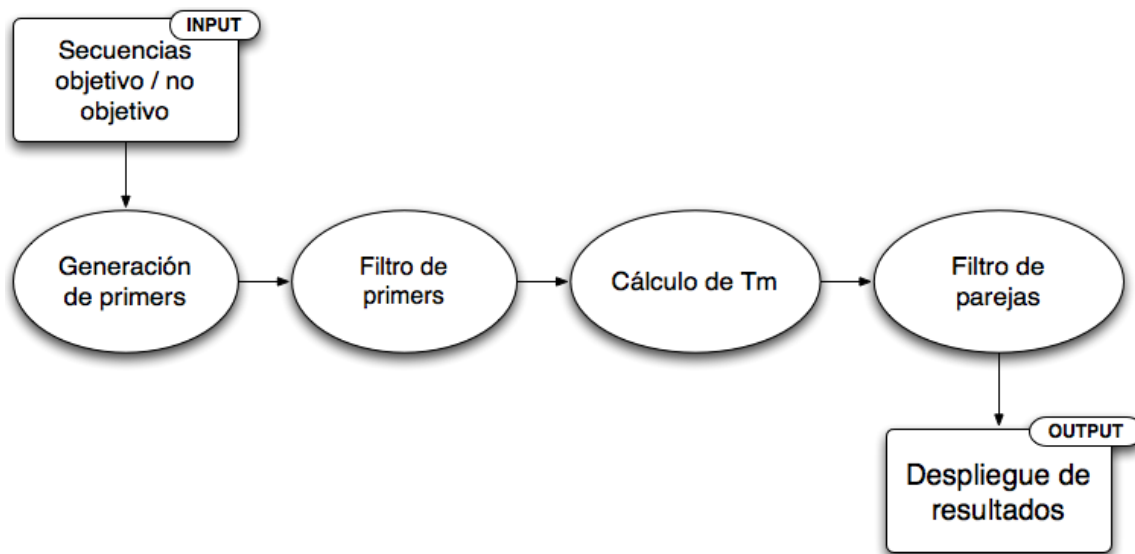


Figura 3.6: Esquema del software PrimerHunter.

El algoritmo funciona de la siguiente manera (ver Figura 3.6): En una primera etapa se generan *primers* a partir de substrings de todas las secuencias objetivo. Posteriormente se filtran los *primers* según satisfagan una formulación matemática de estar dentro de un conjunto definido a partir de parámetros ingresados por el usuario relativos al PCR, los *primers* y el amplicón. Después se calculan las temperaturas de *melting* y los alineamientos termodinámicos óptimos de todas las hibridaciones a evaluar. Finalmente se filtran las parejas de acuerdo a un set adicional de restricciones y se muestran los resultados.

El software está disponible en <http://dna.engr.uconn.edu/software/PrimerHunter/> tanto para bajarlo y correrlo localmente, como para ejecutarlo mediante una interfaz web. Se diferencia de UniPrime y Primers4clades en el sentido que es un software autocontenido, como Pythia, y no un *pipeline* de otras herramientas existentes. Está implementado en C++ y fue validado identificando subtipos de gripe aviar con qPCR's, validando también la precisión de los cálculos de temperatura de *melting*.

De lo revisado en la literatura, es el único trabajo en que se aborda de forma explícita la inclusión de secuencias “no objetivo”, además de proveer formulaciones matemáticas y

técnicas computacionales de utilidad para el problema de diseño de *primers*.

3.3. Requerimientos del Software

Debido a que se espera desarrollar el núcleo de la herramienta en cuanto a funcionalidades de cálculo que pueda tener, junto a requisitos de índole de escalabilidad, estos no son conceptualizables en el esquema de casos de uso. Se tratan principalmente de requisitos de software funcionales y de mantenibilidad de muy alto nivel. Se especificarán como **objetivos específicos** que debe satisfacer la herramienta, según la información provista por los miembros del LBMG. El eje central de las funcionalidades a desarrollar son el de contar con una herramienta efectiva y útil para diseño de *primers* en el LBMG y apuntar en el mediano plazo a capitalizar el trabajo realizado en la forma de una *publicación científica*.

Objetivo específico	Búsqueda versus cálculo
Descripción	Muchos de los cálculos que se hacen sobre las secuencias son costosos -o podrían serlo aun más- y son repetitivos, lo cual hace deseable guardar algunos de estos cálculos de modo de no realizarlos todo el tiempo. Idealmente, pasar a un esquema completo de búsqueda en donde el diseño de <i>primers</i> es una <i>query</i> sobre un universo de secuencias posibles y se retornar las más relevantes según dicha <i>query</i> . Esto es factible teóricamente si el espacio de posibles PCR's se acota, definiendo rangos para los parámetros muy acotados y de tramos discretos, junto con la hipótesis de que muchos de los cálculos realizados para las secuencias de <i>primers</i> se mantienen igual si cambian los universos taxonómicos objetivo.
Prioridad	Media.
Fuente	Andrés Aravena (Ingeniero Jefe LBMG).

Objetivo específico	Paralelización
Descripción	En los recursos de hardware con que cuenta el LBMG está un cluster de computo, brindando la posibilidad de realizar paralelizaciones de los cálculos o las consultas necesarias, por tanto se requiere ocupar dicho recursos paralelizando el proceso en las partes que sea factible y ventajoso de hacerlo.
Prioridad	Alta.
Fuente	Andrés Aravena (Ingeniero Jefe LBMG).

Objetivo específico	Simulación de Hibridación
Descripción	El diseño de <i>primers</i> puede contemplar filtros de los <i>primers</i> y las parejas en función de condiciones necesarias que deben satisfacer, pero también es posible asociar medidas de desempeño de las parejas en función de las secuencias conocidas públicamente, las cuales surgen de simulaciones del proceso de hibridación. De la literatura se pueden vislumbrar aspectos que son mejorables de la simulación de la hibridación y se espera integrar dichas mejoras de forma simple y eficaz al <i>pipeline</i> de trabajo, pues es un área de continuo desarrollo científico. Por tanto, debe ser posible incorporar de forma expedita cambios a las distintas estrategias de simulación que puedan realizarse.
Prioridad	Media.
Fuente	Andrés Aravena (Ingeniero Jefe LBMG).

Objetivo específico	Actualización Base de Datos de secuencias
Descripción	Se estima que el tamaño de las bases de datos de secuencias públicas se duplica cada 18 meses, y por tanto se hace necesario contar con alguna herramienta que automatice el proceso de actualización de la base de datos de secuencias relevantes con las cuales se trabaja constantemente, parametrizable según una política a definir de actualizaciones y taxonomías relevantes.
Prioridad	Alta.
Fuente	Andrés Aravena (Ingeniero Jefe LBMG).

Objetivo específico	Sitio Web
Descripción	Los resultados de los cálculos hechos para el diseño de <i>primers</i> de identificación de ciertos microorganismos relevantes actualmente se despliegan en un sitio web, el cual debe ser rediseñado en función de propiciar la adecuada difusión de la herramienta para diseñar <i>primers</i> en una publicación científica, y brindar una forma ágil de acceder a usar la herramienta para los usuarios del LBMG.
Prioridad	Alta.
Fuente	Andrés Aravena (Ingeniero Jefe LBMG).

Objetivo específico	Mejoras algorítmicas
Descripción	En cada una de las partes del problema de diseño de <i>primers</i> donde se formulan filtros u ordenamientos, los algoritmos que calculan e implementan dichos elementos son susceptibles de ser revisados en su formulación algorítmica con el fin de mejorar la manera en que se realizan y son objeto de continua investigación científica por tanto debe haber flexibilidad en los que se incorporen y factibilidad de incluir nuevos.
Prioridad	Alta.
Fuente	Alejandro Maass (Director LBMG) .

Objetivo específico	Incorporar <i>primers</i> exitosos como input
Descripción	En un esquema de búsqueda versus cálculo o en formulaciones nuevas de los problemas algorítmicos, se espera poder ingresar como input secuencias de <i>primers</i> que se conoce por otros medios como exitosas para realizar alguna amplificación específica.
Prioridad	Media.
Fuente	Marko Budinich (Ingeniero LBMG).

Posteriormente, estas funcionalidades de alto nivel se priorizan para dar paso al diseño e implementación de la herramienta concreta realizada. Para ello se comparan las herramientas y se definen las prioridades para el trabajo final.

3.4. Comparación de las Herramientas

En la revisión de las herramientas similares a la que se busca desarrollar podemos conceptualizar un “*pipeline* generalizado” de lo que significa la tarea de diseñar *primers*. A grueso modo, los pasos que se requieren son:

- **Recibir *Input***: Donde se recibe la o las secuencias que se desean amplificar, además de los parámetros del PCR a realizar, entre otros.
- **Generar *primers***: Obtenidos a partir de algún conjunto definido como subconjunto especificado según los parámetros ingresados por el usuario y las secuencias a trabajar.

- **Ordenar *primers***: Según consideraciones termodinámicas básicas u otras medidas de performance en donde se pueda ordenar los *primers* según dicha medida.
- **Filtrar *primers***: A partir de puntos de corte en función de los parámetros definidos por el usuario o los que ocupen las herramientas.
- **Generar Pares de *primers***: Lo mismo que antes pero considerando el *primer forward* y *reverse*.
- **Ordenar Pares de *primers***: Consideraciones termodinámicas adicionales, entre otras.
- **Filtrar Pares de *primers***: En función de simulaciones u otros criterios.
- **Mostrar Resultados**: En alguna interfaz adecuada, archivos de texto o web.

En virtud de estos pasos, podemos comparar todas las herramientas con la que actualmente se tiene en LBMG y la que se espera desarrollar (ver Cuadro 3.1):

En la columna de *Input* todos reciben la secuencia objetivo que desean amplificar, salvo por LBMG que contempla una selección taxonómica. Se espera poder no sólo ingresar cuáles taxonomías se buscan amplificar sino que también cuáles se desea no amplificar, como lo hace *PrimerHunter*. En la etapa de generación de *primers*, todas consideran o bien substrings de todas las secuencias que reciben (*Pythia*, *PrimerHunter*) o aplican una estrategia de consenso, como *Primers4clades* y *UniPrime*. En el caso de LBMG, se evalúa en la generación misma del *primer* la región de estabilidad SDSS de Miura[19], lo cual se considera se debe mantener así.

El ordenamiento y filtración de *primers* se realiza en las herramientas ya sea delegando el trabajo a CODEHOP (*Primers4clades*) o *Primer3* (*UniPrime*), o bien de acuerdo a diversos parámetros termodinámicos, como lo hace el resto de las herramientas. Una virtud del software *Pythia* es que todos sus cálculos son a partir de consideraciones termodinámicas, por lo que los parámetros de corte para los filtros ingresados por el usuario le son mucho más significativos. Se espera incluir ya sea esto o bien un ajuste de parámetros estadístico a la herramienta a desarrollar en LBMG.

En el trabajo posterior con pares de *primers*, el trabajo sigue delegado a CODEHOP en *Primers4clades* y a *Primer3* en *UniPrime*, el resto de las herramientas considera todos los pares posibles a partir de los *primers* obtenidos en la etapa anterior. Los filtros aplicados a las parejas son todos de índole termodinámica básica, como no hibridación entre *primer forward* y *primer reverse*, entre otros, pero en *Pythia* se incluye además el análisis de la región de estabilidad SDSS, y en LBMG se realiza una simulación de la hibridación con las taxas con las cuales se está trabajando, lo cual se espera mantener y mejorar. En *Primers4clades* se evalúa además el producto amplificado con la herramienta *Amplicon*. *UniPrime* realiza un “PCR virtual” mediante un *blast*. Ambas técnicas eventualmente deseables en el *pipeline* de LBMG.

Cuadro 3.1: Cuadro comparativo de las herramientas revisadas.

	Input	Generación de <i>primers</i>	Ordenamiento de <i>primers</i>	Filtrado de <i>primers</i>	Generación de parejas	Ordenamiento de parejas	Filtrado de parejas	Resultados
Primers4clades	Seq obj	Consenso + Filogenética	CODEHOP	Temperatura de <i>melting</i> (T_m)	CODEHOP	CODEHOP	Amplicon	PhyML
UniPrime	Seq obj + ID Gen	Consenso	Primer3	Primer3	Primer3	Primer3	Elast	Web
Pythia	Seq obj + Locus	Substrings	T_m , Largo del <i>primer</i> y amplificación	Parámetros del usuario	Todos	T_m	Análisis químico + especificidad	Output básico
PrimerHunter	Seq obj + seq no obj	Substrings	GC, estabilidad, etc	Parámetros del usuario	Todos	Largo del <i>primer</i> , T_m , SDSS	Parámetros del usuario	Web
LBMG actual	Taxa	SDSS	Varios termodinámicos	Parámetros del usuario	Todos	Simulación hibridación	Parámetros del usuario	Web
LBMG futuro	Taxa obj + Taxa no obj	SDSS	Varios termodinámicos	Parámetros del usuario + ajuste	Todos	Simulación hibridación	Parámetros del usuario + ajuste	Web ++

Finalmente, los resultados se muestran en una página web, y su visualización se puede enriquecer filogenéticamente como en el caso de **Primers4clades** mediante **PhyML**. En LBMG se espera realizar una mejora al sitio web que permita mostrar las ventajas del *pipeline* final (de manera informativa), además de permitir enriquecer la visualización de los datos con herramientas externas.

3.5. Estrategia de Desarrollo y Ambiente Operacional

Revisadas las herramientas a considerar, existe una serie de decisiones que tomar en función de los requerimientos, sus prioridades y los *trade-offs* involucrados en considerar una alternativa frente a otras. El primer grupo de decisiones es ordenar los requisitos para implementar las características de software necesarias de forma iterativa. Posteriormente se debe decidir entre realizar un *pipeline* o un software completo. Finalmente, se debe decidir la metodología y ambiente operacional para implementar la solución.

Cualitativamente, podemos ordenar los requerimientos en una tabla de doble entrada, y en función del objetivo principal que es obtener de este trabajo una herramienta efectiva y útil, determinar si la materialización de un requerimiento beneficia a otro o no (ver Cuadro 3.2):

Así, se desprende como necesario la realización de al menos la actualización de las bases de datos automática y un sitio web acorde a lo que se espera divulgar en la comunidad científica. Como la herramienta se hace cargo de varias *features* que no están directamente relacionadas con el trabajo futuro, sino más bien constituir su base, el resto de los requerimientos se priorizó en función de como se llevó a cabo el trabajo de memoria.

Para poder realizar las extensiones a la herramienta, se necesita contar con un rediseño e implementación de la misma que permita hacerlo de forma eficiente y comunicable. Por ello, se realizó un diseño orientado a objetos del software, el cual en algunos aspectos funciona como un *pipeline* puesto que se trabaja de forma explícita con herramientas como **blast** o **Unafold**[18] para externalizar ciertas tareas.

Para validar el diseño y su posibilidad de extensión, se propone una serie de extensiones específicas y su forma de integrarlas a la herramienta desarrollada. El ambiente operacional para desarrollar la herramienta son los servidores del LBMG.

Cuadro 3.2: Requerimientos y su impacto entre sí.

	Pub	BvsC	Par	SimHib	ActBD	Web	Alg	<i>Primers</i>
Publicación (Pub)		-	-	+	+	+	+	±
Búsqueda Vs Cálculo (BvsC)			+	-	+	-	+	+
Paralelización (Par)				-	+	-	+	-
Simulación Hibridación (SimHib)					-	-	+	-
Actualización Base de datos (ActBD)						+	-	+
Sitio Web							-	+
Mejoras Algorítmicas (Alg)								-
<i>Primers</i> Input								

3.6. Comparación de sistema anterior y nueva formulación

Para comprender el funcionamiento del sistema existente, se debió realizar una exhaustiva revisión junto a los ingenieros del LBMG, y se creó documentación para el sistema antigua, que no existía. A continuación, se presenta la parte del preprocesamiento (ver Figura 3.7):

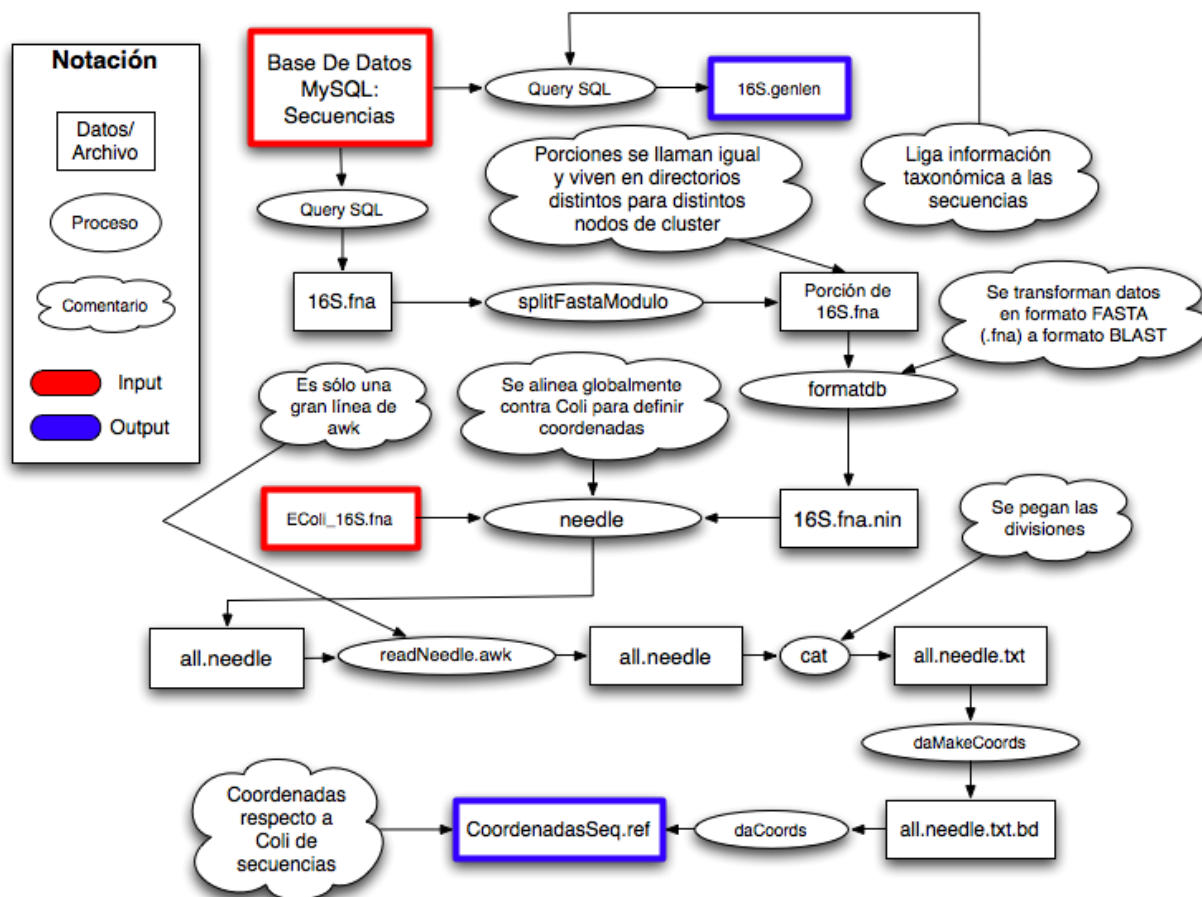


Figura 3.7: Esquema del pre proceso del sistema actual.

En síntesis, se obtienen secuencias según el universo taxonómico definido de forma puntual en archivos `makefile` adecuados, desde la base de datos de secuencias local y posteriormente se calculan o procesan distintas bases de datos de texto para implementar la estrategia del laboratorio y paralelizar los cálculos.

El proceso de filtrado de las secuencias se expone en la siguiente figura (ver Figura 3.8):

El flujo es como sigue: Se generan *primers* con la estrategia SDSS y filtrando por largo,

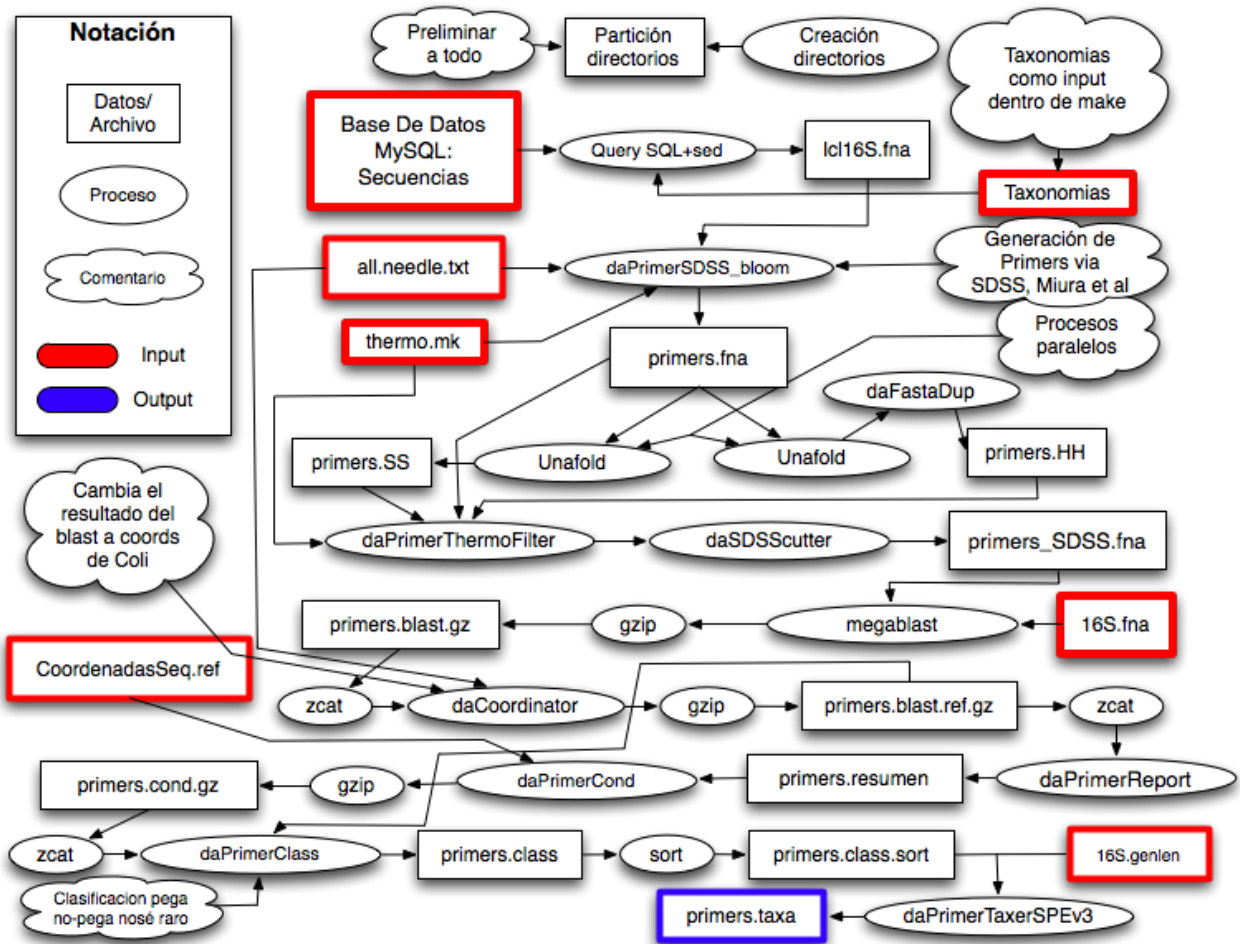


Figura 3.8: Esquema del proceso de filtros actual.

temperatura, contenido GC, entre otros, para posteriormente aplicar otros filtros termodinámicos, como de hibridación de los *primers* consigo mismos, entre otros. Finalmente se llega a un conjunto de *primers* apto para formar parejas que se filtran de forma análoga. La idea es que esto es conceptualizable de forma más ordenada como se sigue (ver Figura 3.9):

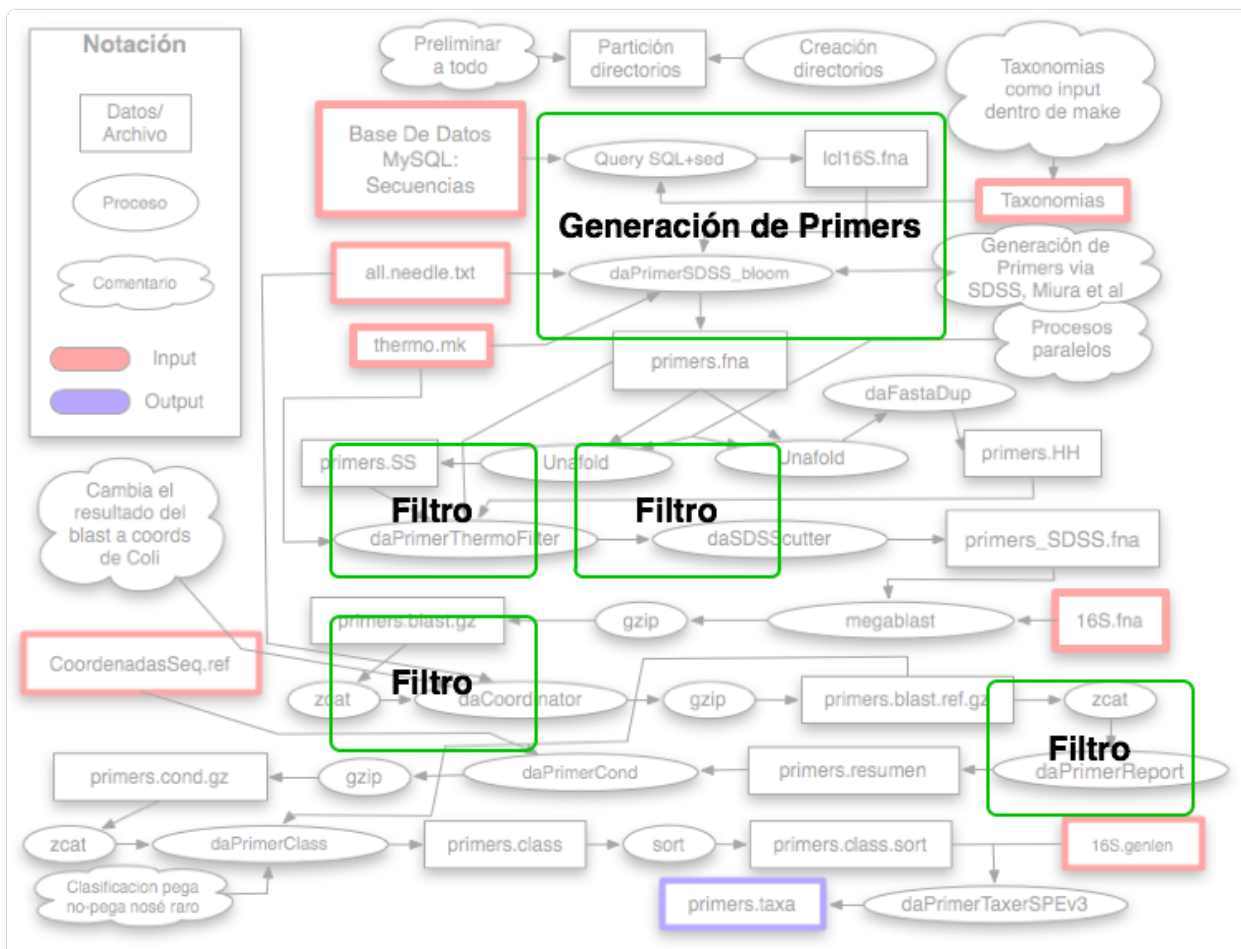


Figura 3.9: Esquema del proceso anterior bajo la nueva conceptualización.

De esta forma, podemos abordar los distintos objetivos específicos de la siguiente manera:

- El esquema de búsqueda versus cálculo requiere realizar pruebas intensivas para corroborar las hipótesis que deben sustentarlo, y para ello, un sistema que se opera de forma efectiva y que está suficientemente bien ordenado y comunicable, permite realizar dichas tareas expeditamente.
- La paralelización en el sentido de esta aplicación consiste en distribuir cálculos, lo cual separando el preproceso y partición de los datos de los cálculos que se van haciendo, garantiza que dicho aspecto se pueda trabajar de forma adecuada.

- Toda simulación de hibridación sería desarrollar y testear un filtro más, al igual que la mayoría de los filtros aplicados, que operando con una interfaz explícita permite trabajar con ellos de forma independiente. Esto se extiende de la misma manera a las mejoras algorítmicas específicas.
- La actualización de las bases de datos de secuencias es un problema atingente al sistema completo pero no directamente relacionado al re diseño de la herramienta, por lo tanto se delimita como fuera del alcance de este trabajo de memoria.
- El sitio web aplica dentro de la misma categoría del objetivo anterior, por tanto se baja su prioridad del mismo modo que el anterior.
- La incorporación de los *primers* exitosos es un objetivo que se puede abordar de forma localizada en la generación de los *primers*, simplemente desarrollando un nuevo criterio de generación.

De esta forma, se concluye que la conceptualización, re diseño y re implementación propuestos y realizados en esta memoria permiten conformar el **núcleo** de la herramienta que permitirá abordar los desafíos expuestos como objetivos específicos.

Capítulo 4

Diseño del Sistema

4.1. Arquitectura del Sistema

4.1.1. Componentes

La Arquitectura del sistema la podemos definir para efectos de este diseño como la visión global de sus componentes. En las secciones previas se delinearon los rasgos que debiese tener un sistema flexible y escalable de diseño de *primers*, los cuales procederemos a detallar a continuación en función de componentes, principios y patrones de diseño. Esta descripción busca ser además una descripción lo más amplia posible para exponer el sistema a quienes necesiten trabajar y operar con él.

Las componentes del sistema son:

1. **PrimerConstants:** En este módulo se guardan todas las constantes que deben estar definidas para el resto de los módulos y sus respectivas clases y funciones tales como la constante de los gases para cálculos termodinámicos, umbrales de corte de atributos por defecto, valores de entrada por defecto, etc.
2. **PrimersGenerators:** En este módulo se distingue el proceso de filtrado de un conjunto de *primers* del de generación de los mismos. Esto es necesario pues a priori, *todos* los substrings de una secuencia dada son candidatos a ser *primers*, pero por motivos de eficiencia no es adecuado mantenerlos todos para hacer filtros posteriores directos, como rangos de largo de los *primers* o que satisfagan tener alguna propiedad termodinámica básica, como una temperatura de *melting* en un rango dado. Es por ésto que se mantienen en éste módulo.
3. **ObjectsCollection:** Aquí se encuentran todas las clases de los objetos que otros módu-

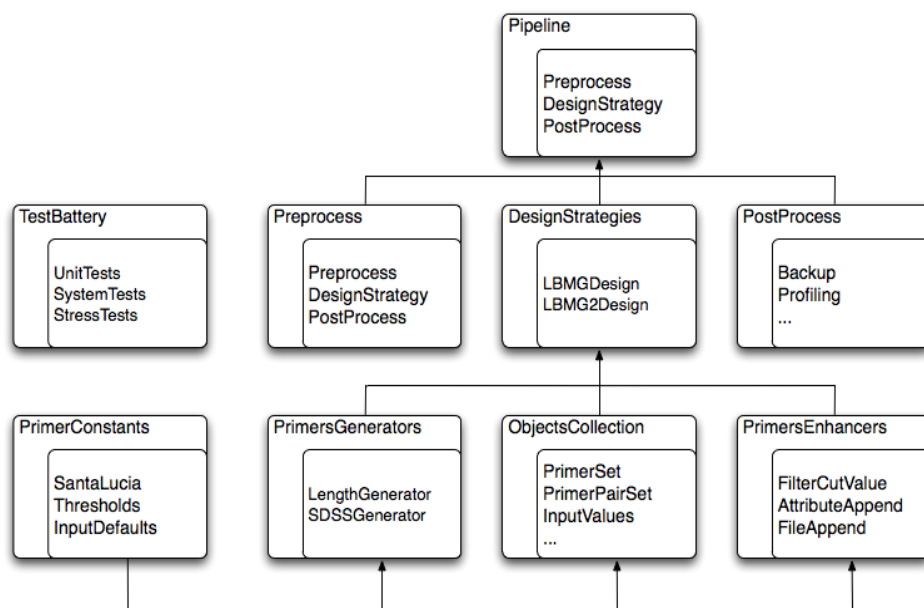


Figura 4.1: Esquema de las componentes del sistema

los incorporan como atributos en sus respectivas clases. Los objetos más importantes en éste módulo son `PrimerSet` y `PrimerPairSet`, los cuáles contienen un conjunto de secuencias y *metadata* asociada a cada conjunto, *primers forward* y *primers reverse* el primero, y pares de *primers forward* y *primers reverse* el segundo.

4. `PrimersEnhancers`: El conjunto de filtros, o en el sentido amplio, “mejoradores” de `PrimerSet` y `PrimerPairSet`¹. Las clases aquí presentes pueden clasificarse en general cómo:

- `FilterCutValue`: Una clase para filtrar un conjunto de *primers* de acuerdo a un valor umbral sobre un atributo de los *primers*.
- `AttributeAppender`: Clases que añaden atributos a los *primers* en función de otros atributos.
- `FileAppender`: Clases que asocian un archivo al conjunto, en general una base de datos de texto con datos relacionando los *primers* con otro conjunto de secuencias.

5. `TestBattery`: Conjunto de pruebas para realizar sobre el sistema.

¹ Por ello el nombre *Enhancers*. Todo el código fue descrito y nombrado en inglés para contar con un sistema que sea rápidamente inspeccionable por una amplia gama de usuarios de la comunidad científica con la cual trabaja el LBMG.

6. **Preprocess:** Módulo con las tareas que se deben realizar antes de ofrecer opciones de entradas para el sistema.
7. **DesignStrategies:** Aquí están los contenedores para las estrategias que se creen para el diseño de *primers*.
8. **PostProcess:** Módulo con las tareas que se deben realizar posteriormente a una corrida del sistema.
9. **Pipeline:** El *pipeline* de diseño completo de *primers*.

4.1.2. Dinámica del Sistema

En el momento de ejecutar una instancia para el sistema, las diversas componentes del mismo se relacionan de forma específica para producir un resultado. El flujo lo podemos describir en los siguientes pasos:

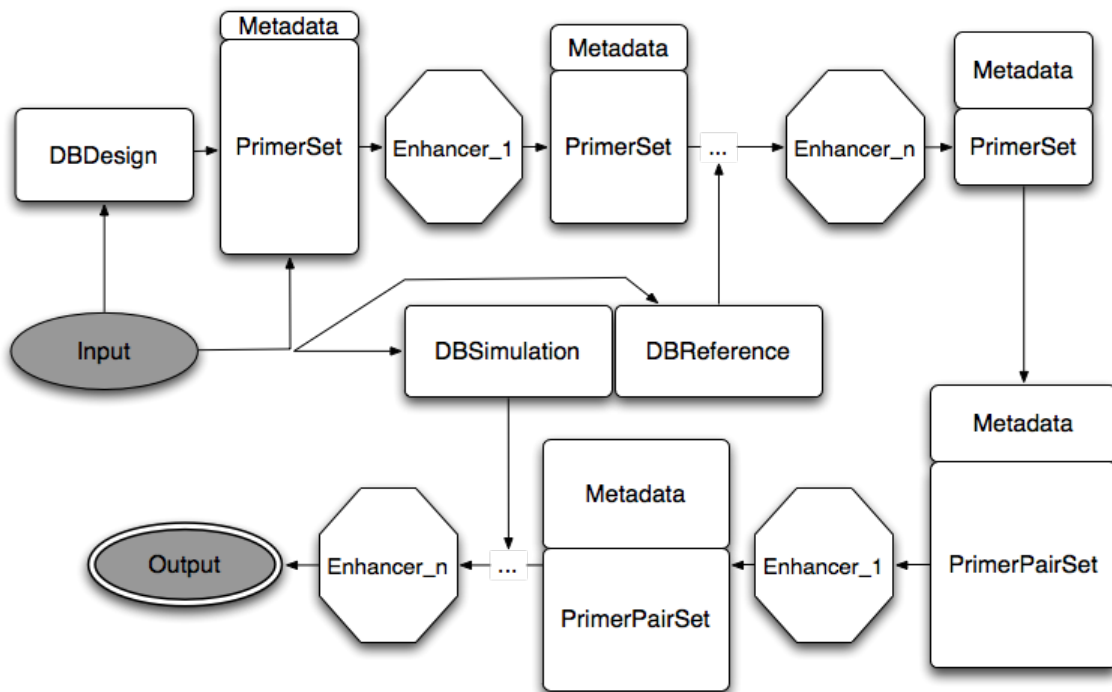


Figura 4.2: Esquema del flujo del sistema

- **Ingreso de *input*:** En una primera etapa se ingresa parámetros de entrada para un diseño particular de *primers*. Luego se instancian tres bases de datos específicas para dicha entrada, que son **DBDesign** (Base de datos de diseño) a partir de la cual se obtienen subsecuencias que son candidatos a *primers*, **DBReference** (Base de datos de referencia) que contiene las secuencias que se espera estén en la muestra metagenómica a trabajar y **DBSimulation** (Base de datos de simulación) que puede ser igual a la base de datos de referencia, pero con un conjunto reducido de secuencias, pues el proceso de simulación es costoso computacionalmente.
- **Creación de PrimerSet:** Se instancia un objeto **PrimerSet**, el cual contiene un conjunto en un principio grande de *primers*, según el tamaño de la base de datos de diseño y el algoritmo escogido para la generación. También se asocia al **PrimerSet** un conjunto de *metadata*, como los archivos con las bases de datos descritas, entre otras cosas.
- **Aplicación de Enhancers:** En esta etapa, se aplica una serie de filtros, añadidores de atributos y asociadores de archivos, que van reduciendo el conjunto de *primers*, y eventualmente acrecentando la *metadata* asociada. En particular algunos **Enhancers** pueden usar las bases de datos asociadas.
- **Creación de PrimerPairSet:** A partir del **PrimerSet** resultante de los pasos anteriores, se crea un conjunto de pares de *primers forward* y *primers reverse*, así como también *metadata* respectiva.
- **Aplicación de Enhancers:** Del mismo modo que en el filtrado de **PrimerSet**, se aplican **Enhancers** a **PrimerPairSet**, con el mismo propósito de reducir el conjunto y asociar *metadata*. En una de estas etapas se realiza una simulación de la hibridación de las parejas de *primers* con la base de datos de simulación.
- **Obtención de *output*:** Finalmente los resultados obtenidos se post procesan y se presentan al usuario.

4.1.3. Lenguaje de desarrollo

Una decisión de diseño importante que es necesario abordar en este punto es el lenguaje de programación a usar para desarrollar el sistema. Las alternativas consideradas fueron C++, Perl, Python y Java. Perl y Python son ampliamente usados en la comunidad bioinformática, estando ambos presentes en una gran cantidad de herramientas del área. Java y C++ fueron considerados en función de su amplia presencia en contextos empresariales y científicos, respectivamente. Finalmente, se evaluó como un aspecto muy relevante la mantenibilidad del código y que el “costo de entrada” para modificar el sistema fuese bajo, sujeto a que fuera un lenguaje conocido en la comunidad científica y bioinformática. Por estas razones, sumado a una serie de ventajas técnicas específicas del lenguaje, se optó por **Python**.

Las características particulares que hacen de Python un lenguaje deseable son: accesibilidad a amplia documentación, variadas librerías y *frameworks* para resolver diversas necesidades recurrentes, soporte de paradigmas de orientación a objetos y funcional, extensibilidad, escalabilidad y el poseer un intérprete interactivo que permite por un lado realizar prototipos de manera rápida y por otro, brinda la posibilidad a futuro de convertir el trabajo de esta memoria en una librería de diseño de *primers*.

4.1.4. Patrones de diseño considerados

En el diseño detallado del sistema se abordó la identificación de secciones en las cuales fuese conveniente usar las conceptualizaciones de los patrones de diseño de software. Los patrones por cuya aplicabilidad y relevancia fueron considerados son:

- **Template Method:** Por la frecuencia con que algunos pasos del diseño de *primers* son abstraibles y para proveer un cierto control en la forma en que se ejecutan dichas abstracciones, se considera necesario y pertinente el uso de éste patrón en el diseño.
- **Strategy:** Debido a que dependiendo de la instancia y/o aplicación específica, algunos algoritmos pueden ser más apropiados que otros para llevar a cabo tareas como la generación de *primers*, simulaciones de hibridación u otros, es deseable conceptualizar el diseño de esas secciones del sistema con éste patrón.
- **Facade:** Para facilitar el uso de conjuntos de filtros recurrentes, se considera valioso utilizar este patrón para diseñar objetos contenedores con una interfaz simplificada.
- **Decorator:** Para brindar extensibilidad de los objetos que fruto de revisión de lo que modelan, podrían requerir ser extendidos en una forma que no es anticipable a priori y que procure mantener las interfaces ya definidas.

4.2. Diseño detallado y evaluación

En esta sección se aborda el diseño detallado de las clases del sistema. Para ello, se discute y expone los tipos de usuarios, los módulos implementados y el alcance de lo realizado, los criterios de evaluación del diseño y descripciones para cada clase de acuerdo a su interfaz, junto al proceso de diseño que le precedió y su evaluación.

Los usuarios del sistema en orden creciente de *features* que pueden acceder son:

- **Diseñadores de *primers*:** Son los usuarios que requieren diseñar *primers* en meta-genómica para una aplicación específica. Se espera que definan sus criterios de diseño

a través de especificar valores en un formulario de entrada, usando varios valores para el diseño y alguna estrategia por defecto.

- **Diseñadores de estrategias:** Usuarios que buscan explorar nuevas estrategias de diseño, a través del uso de los **Enhancers** en distintos ordenes e instanciándolos de maneras distintas a las que se consideran por defecto, para después emplear la estrategia en forma sistemática por parte de diseñadores de *primers*.
- **Desarrolladores:** Son los usuarios que extenderán el sistema, especialmente mediante la creación de nuevos **Enhancers** u otros objetos para incluir nuevas estrategias, emular otras herramientas de diseño de *primers*, ampliar los contextos de aplicación a esquemas de diseño de secuencias para otras aplicaciones, etc.

El alcance del desarrollo abordado en este trabajo de memoria se centra en poder instanciar de la forma más completa posible el *pipeline* de diseño que ya existe actualmente, proveyendo correcciones y mejoras donde sea directo hacerlo. En función de ello, los módulos con los que se trabajó para portar el trabajo ya existente son **PrimerConstants**, **ObjectsCollection**, **Preprocess**, **PrimerGenerators** y **DesignStrategies**. Los otros módulos de la arquitectura definida están vinculados a las extensiones a realizar de la herramienta.

Los criterios de evaluación del diseño de clases son:

- Las clases se definen mediante una descripción general, sus atributos y principalmente, su **interfaz**.
- El comportamiento que define la interfaz debe ser sometido a las siguientes preguntas:
 - ¿Es relevante? ¿Demandado?
 - ¿Es complejo? ¿Factible?
 - ¿Es dependiente de su implementación?
 - ¿Es útil en otro contexto?
- En la evaluación de cada clase se discutirá cuáles fueron las alternativas consideradas para decidir la especificación final de la clase.

4.2.1. PrimerConstants

En este módulo se definen las constantes y valores por default que otras clases usan. Incluye valores por defecto para *inputs* y constantes que definen atributos de algunos objetos. Por ejemplo:

Valores de <i>input</i> por defecto	<pre> PCRConcentrations_monovalentCation_DEFAULT=50E - 3 PCRConcentrations_dNTPs_DEFAULT=0,0 ... ThermoDynamicsParams_PCRTemperature_DEFAULT=50,0 </pre>
Constantes de atributos para objetos	<pre> CUT_OFF_VALUE_THRESHOLD='Threshold' SANTALUCIA_CONSTANTS_DICT['GAS_CONSTANT']=1,987 ... SANTALUCIA_CONSTANTS_DICT['GC_ENTHALPY']=0,1 </pre>

Como se trata de un módulo que solo tiene constantes, sin definir clases ni métodos, la evaluación definida no aplica a este módulo.

4.2.2. ObjectsCollection

Éste módulo contiene los objetos que tienen presencia en varios módulos, en particular los objetos centrales de este diseño: `PrimerSet` y `PrimerPairSet`.

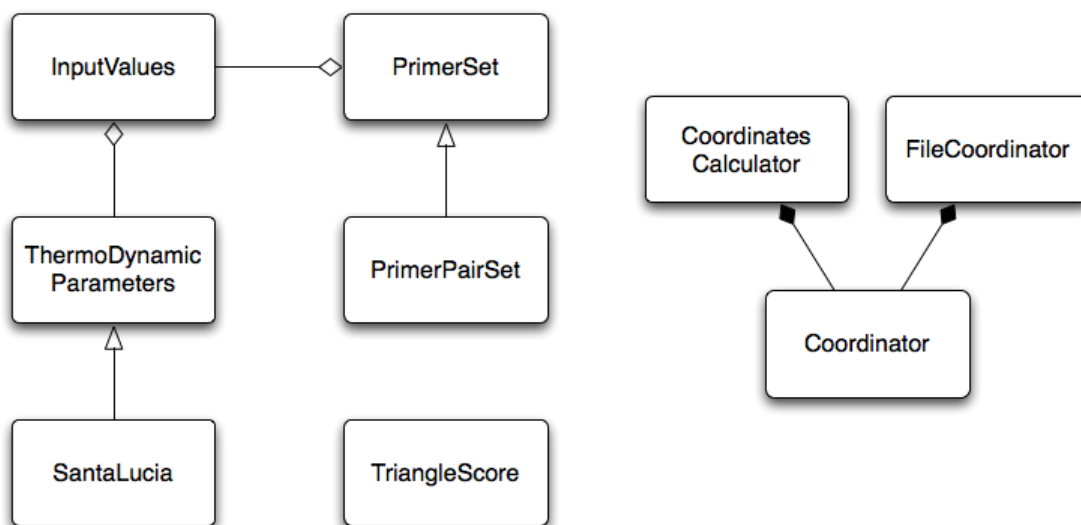


Figura 4.3: Diagrama de clases del módulo

El esquema general de las clases (ver Figura 4.3) de este módulo es: La clase `SantaLucia`

hereda de `ThermoDynamicParameters`, `InputValues` contiene una instancia de este último objeto, y las instancias de `InputValues` a su vez son contenidas en `PrimerSet` y `PrimerPairSet`. `TriangleScore` permite calcular puntajes de forma “triangular”, y finalmente `Coordinator` es contenido en las clases `CoordinatesCalculator` y `FileCoordinator`, que proveen distintas variantes para la funcionalidad provista esencialmente por `Coordinator`.

Cuadro 4.1: Clase `ThermoDynamicParameters`

Nombre clase	<code>ThermoDynamicParameters</code>
Descripción	Contiene los valores de entropía y entalpía para cálculos termodinámicos relacionados con secuencias, entre otros.
Atributos	<ul style="list-style-type: none"> ▪ <code>entropy</code>: Un diccionario con valores de entropía para pares de bases como llaves. ▪ <code>enthalpy</code>: Análogo a <code>entropy</code> pero con valores de entalpía. ▪ <code>constants</code>: Análogo a los anteriores, pero con valores como la constante de los gases (R). ▪ <code>correctionMethod</code>: El nombre del método de corrección termodinámica.
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>: Inicializa una instancia a partir de valores por defecto. ▪ <code>getEntropy(key)</code>: Retorna el valor de la entropía para un par de bases (<code>key</code>), si el par no está presente, retorna <code>None</code>. ▪ <code>getEnthalpy(key)</code>: Retorna el valor de la entalpía para un par de bases (<code>key</code>), si el par no está presente, retorna <code>None</code>. ▪ <code>entropyBeginEnd(base)</code>: Retorna el valor de la entropía para una sola base. Si la base no está presente, retorna <code>None</code>. ▪ <code>enthalpyBeginEnd(base)</code>: Retorna el valor de la entalpía para una sola base. Si la base no está presente, retorna <code>None</code>. ▪ <code>saltCorrectionOnline(deltaEntropy, length, salt)</code>: Retorna el valor de corrección de entropía con la sal (Na^+) presente en una reacción. ▪ <code>__str__</code>: Retorna una descripción de la instancia.

Cuadro 4.2: Clase `SantaLucia`

Nombre clase	<code>SantaLucia</code>
--------------	-------------------------

Continúa en página siguiente...

Cuadro 4.2 – Continuación

Descripción	Parámetros termodinámicos según SantaLucia et al.[27]. Extiende la clase <code>ThermoDynamicParameters</code> , definiendo explícitamente los métodos <code>__str__</code> y <code>saltCorrectionOnline</code> .
Atributos	<ul style="list-style-type: none"> ▪ Los que hereda de la clase <code>ThermoDynamicParameters</code>.
Métodos	<ul style="list-style-type: none"> ▪ Los que hereda de la clase <code>ThermoDynamicParameters</code>.

Cuadro 4.3: Clase `InputValues`

Nombre clase	<code>InputValues</code>
Descripción	El objeto que contiene los atributos para un <i>input</i> del sistema, con parámetros específicos a un diseño de <i>primers</i> puntual. Para efectos del desarrollo realizado en este trabajo de memoria, los valores se inicializan por defecto, considerando en el futuro tener un constructor y validador conectado con las interfaces de usuario a desarrollar.

Continúa en página siguiente...

Cuadro 4.3 – Continuación

Atributos	<p>Por motivos de legibilidad describimos sólo los atributos utilizados en el sistema desarrollado para esta memoria y en conjuntos:</p> <ul style="list-style-type: none"> ▪ queryID: Un identificador para la instancia particular de diseño de <i>primers</i>. ▪ PCRConcentrations: Concentraciones para cationes monovalentes, divalentes, <i>primers</i>, bases e iones de sodio presentes en la solución del PCR. ▪ ThermoDynamics: Constantes y métodos termodinámicos, en particular una instancia del objeto ThermoDynamicParameters. ▪ Constrains: Diversos conjuntos de restricciones, en general en listas, con mínimos, máximos y opcionalmente óptimos, para las temperaturas de <i>melting</i>, contenido GC, largo del producto amplificado, largo de los <i>primers</i>, etc. ▪ Target_Sequence: Datos adicionales asociados a secuencias objetivo. ▪ Non_Target_Sequence: Datos adicionales asociados a secuencias no objetivo. ▪ Target_taxIDs: Identificadores de las taxas objetivo. ▪ Non_Target_taxID: Identificadores de las taxas no objetivo.
Métodos	<ul style="list-style-type: none"> ▪ __init__: Inicializador para las instancias de la clase. Recibe un parámetro opcional (por defecto vacío) para retomar los valores de <i>input</i> a partir de un archivo. ▪ __str__: Descriptor. ▪ __getitem__(key): Posibilita tratar los atributos del objeto como llaves, y al objeto como un diccionario.

Cuadro 4.4: Clase TriangleScore

Nombre clase	TriangleScore
---------------------	---------------

Continúa en página siguiente...

Cuadro 4.4 – Continuación

Descripción	Clase que permite calcular un puntaje para un valor situado en un rango con algún valor opcional óptimo. El puntaje es lineal en el rango con valor máximo en el punto óptimo.
Atributos	<ul style="list-style-type: none"> ▪ xMin: Valor mínimo para el rango. ▪ xMax: Valor máximo para el rango. ▪ xOpt: Valor óptimo para el rango. ▪ Max: Valor máximo del puntaje, por defecto 1,0. Mínimo es 0.
Métodos	<ul style="list-style-type: none"> ▪ __init__: Inicializa una instancia de la clase. ▪ __str__: Retorna una descripción del objeto. ▪ __call__(x): Posibilita que el objeto sea llamado como una función.

Cuadro 4.5: Clase PrimerSet

Nombre clase	PrimerSet
Descripción	La clase que contiene los <i>primers</i> generados, archivos y otros datos asociados a él. Los <i>primers</i> se guardan en una matriz, con <i>primers</i> como líneas y atributos como columnas.

Continúa en página siguiente...

Cuadro 4.5 – Continuación

Atributos	<ul style="list-style-type: none">■ setName: Un nombre para la instancia de la clase, semánticamente descriptivo, obtenible desde un <i>input</i> específico para el sistema.■ input: Una instancia de la clase <code>InputValues</code>.■ attribList: Un diccionario con nombres de atributos como llaves e índices de columnas en la matriz de <i>primers</i> como valores.■ seqList: La matriz de <i>primers</i>.■ enhancersList: Una lista con los objetos <code>Enhancers</code> aplicados al conjunto, mediante descripciones que dejan cada uno en esta lista y llevar así una historia de lo que ha sucedido con la instancia.■ filesDict: Un diccionario con palabras clave como llaves y <i>paths</i> de archivos como valores. Usado para guardar los archivos con <i>metadata</i> asociada.■ recoverDict: Un diccionario similar al anterior, pero creado en el momento de realizar una operación de <code>Backup</code> a una instancia del objeto.
------------------	---

Continúa en página siguiente...

Cuadro 4.5 – Continuación

Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>: Inicializa una instancia de la clase. Puede ser inicializada con valores <code>None</code> para después realizar un <code>recover</code>. ▪ <code>__str__</code>: Retorna una descripción del objeto, llamando a <code>getStatus</code> y <code>getPrimers</code>. ▪ <code>getStatus</code>: Retorna una descripción del objeto indicando el contenido de sus atributos <code>input</code>, <code>enhancersList</code>, <code>filesDict</code> y <code>seqList</code>, pudiendo omitir los <i>primers</i>. ▪ <code>getFasta</code>: Retorna un <code>string</code> en formato <code>fasta</code> de las secuencias de los <i>primers</i>, recibiendo como parámetro opcional que atributos incluir como <i>metadata</i>. ▪ <code>getPrimers</code>: Retorna un <code>string</code> en formato de columnas separadas por <code>tabs</code>. ▪ <code>getEnhancers</code>: Retorna una descripción de los <code>Enhancers</code> aplicados a la instancia. ▪ <code>recover</code>: Posibilita que el objeto sea recuperado a partir de un conjunto de archivos. Pueden estar ya en el objeto, o bien puede proveerse un diccionario opcional, que satisfaga tener las llaves requeridas para la operación.
---------	--

Cuadro 4.6: Clase `PrimerPairSet`

Nombre clase	<code>PrimerPairSet</code>
Descripción	Clase que hereda de <code>PrimerSet</code> , pero que en lugar de tener una matriz de <i>primers</i> , tiene una matriz de pares de <i>primers</i> .
Atributos	<ul style="list-style-type: none"> ▪ Los atributos que hereda de <code>PrimerSet</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>: Inicializa una instancia de la clase. Recibe un objeto <code>PrimerSet</code> y un nombre para construir las parejas. Opcionalmente puede recibir un diccionario para nuevos mapeos de las llaves del atributo <code>filesDict</code> de dicho <code>PrimerSet</code> a nuevas llaves. ▪ El resto de los métodos heredan de <code>PrimerSet</code>.

Cuadro 4.7: Clase CoordinatesCalculator

Nombre clase	CoordinatesCalculator
Descripción	Calcula la posición de una base nucleotídica de una secuencia <i>A</i> en una secuencia <i>B</i> , dadas ambas secuencias alineadas.
Atributos	<ul style="list-style-type: none"> ▪ p: Un objeto patrón (de expresión regular) utilizado para buscar en las secuencias alineadas. ▪ gapChar: Un carácter que representa <i>gaps</i> en las secuencias alineadas. ▪ fictitiousChar: Un carácter de reemplazo para calcular las posiciones.
Métodos	<ul style="list-style-type: none"> ▪ __init__: Inicializa una instancia de la clase. ▪ __str__: Retorna una descripción del objeto. ▪ __call__(A,B,x): Retorna una posición <i>x</i> de <i>A</i> en <i>B</i>.

Cuadro 4.8: Clase FileCoordinator

Nombre clase	FileCoordinator
Descripción	<p>Toma un archivo modificado de resultados de alineamiento vía el programa <i>needle</i>^a, en que cada resultado del alineamiento está en una sola línea, y transforma la posición inicial y final de la segunda secuencia alineada en las “coordenadas” de la primera secuencia, según se define en el alineamiento.</p> <hr style="width: 20%; margin-left: 0;"/> <p>^aVer sección implementación</p>
Atributos	<ul style="list-style-type: none"> ▪ calc: Un objeto CoordinatesCalculator. ▪ inputFilename: El <i>path</i> al archivo que contiene el alineamiento modificado.
Métodos	<ul style="list-style-type: none"> ▪ __init__: Inicializa una instancia de la clase. ▪ __call__(outputFilename): Realiza el cálculo y lo escribe a un archivo que recibe de argumento.

Cuadro 4.9: Clase Coordinator

Nombre clase	Coordinator
---------------------	-------------

Continúa en página siguiente...

Cuadro 4.9 – Continuación

Descripción	Toma un archivo modificado de resultados de alineamiento vía el programa <code>needle</code> , guarda todas las líneas de dicho archivo en un diccionario para calcular los cambios de coordenadas rápido y en demanda.
Atributos	<ul style="list-style-type: none"> ▪ <code>calc</code>: Un objeto <code>CoordinatesCalculator</code>. ▪ <code>referenceSeqs</code>: Un diccionario con las secuencias de referencia alineadas, con los nombres de las secuencias alineadas con ellas como llaves. ▪ <code>alignedSeqs</code>: Un diccionario con las secuencias alineadas, con los nombres de dichas secuencias como llaves.
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>: Inicializa una instancia de la clase. ▪ <code>__call__(seqName, x)</code>: Retorna la coordenada x de <code>seqName</code> en su respectiva secuencia alineada de referencia.

Evaluación

La clase `ThermoDynamicParameters` abstrae todos los parámetros necesarios para cálculos termodinámicos, independiente del modelo a usar. La clase `SantaLucia` particulariza algunos de estos parámetros según el modelo de SantaLucia, brindando de modo directo la posibilidad de extenderlo a nuevos modelos. Estos contenedores son relevantes, demandados y útiles en muchos cálculos de atributos y simulaciones termodinámicas de las secuencias. Como son objetos esencialmente contenedores no son dependientes de su implementación.

La clase `InputValues` actúa a su vez como contenedora de todos los parámetros necesarios para tratar una instancia particular del diseño de *primers*. Su comportamiento está en función de su labor de contenedora y encapsuladora de los atributos necesarios para un *input*. Es una clase ampliamente utilizada en los filtros y `Enhancers` que requieren de datos específicos para instanciarse. Como los objetos previamente discutidos, tampoco es dependiente de su implementación. Eventualmente se consideró aplicar el patrón de diseño `Decorator` que permite extender dinámicamente una clase, lo cual se descartó a posteriori por ser implementado de forma directa en Python por ser un lenguaje de tipado dinámico.

Los objetos `PrimerSet` y `Primer-Pair-Set` fueron concebidos inicialmente de forma separada, sin embargo luego de ir implementándolos se consideró hacer que el segundo heredara del primero, pues estructuralmente son iguales y la distinción es sólo semántica sobre los datos, y por ello se crea un nuevo objeto `PrimerPairSet` cuya principal distinción es la forma

de construirse a partir de un objeto `PrimerSet`. El comportamiento de este objeto es central en el diseño, pues conceptualiza todo lo necesario para tener un objeto flexible, extensible y completo para lo que se espera realizar en la tarea de diseñar *primers*. Los tipos de datos de sus atributos y la implementación de sus métodos son fácilmente portables a varios lenguajes o modificables. Una modificación directa al diseño para abordar en el futuro es incorporar la función de `Backup` (actualmente considerada un `Enhancer`) dentro de la interfaz de `PrimerSet` y hacer que éste objeto sea más autocontenido.

El modelo de “Cambio de coordenadas” implementado por el trío de clases `Coordinator`, `CoordinatesCalculator` y `FileCoordinator` permiten considerar un alineamiento entre dos secuencias como una forma de escribir posiciones de una base de una secuencia en otra de referencia, permitiendo así comparar posiciones de alineamiento de los *primers* entre diversas secuencias de taxás eventualmente distintas o divergentes. Se sustenta en el paradigma de contar con organismos modelo dentro de un subárbol del árbol taxonómico de las especies. `Coordinator` abstrae lo necesario para realizar los cálculos de cambio de coordenadas y las otras dos clases permiten hacer dichos cambios para un archivo completo o guardarlo para realizarlo en demanda. Como sirven el propósito de hacer comparables posiciones de alineamiento de distintas secuencias, son comportamientos relevantes en varias secciones del sistema y son implementables de manera directa con cualquier lenguaje que permita manejar expresiones regulares.

Después de un rediseño, se puede considerar que `PrimerSet` y `PrimerPairSet` son dos instancias de una misma clase, incorporación que se considera para trabajo futuro. También cabe destacar que los formatos de los archivos (como `fasta`) debiesen ser parámetros y no métodos separados.

4.2.3. Preprocess

Las tareas que se llevan a cabo como pasos previos a instancias sistemáticas de diseño de *primers* se consideran acá. Para un usuario diseñador de *primers*, las clases que se presentan a continuación son parte de un preprocesamiento. Para los desarrolladores y diseñadores de estrategias estas tareas pueden ser parte de las tareas que esperan realizar y por tanto requerirían que más abstracciones y nuevas clases que se hacen cargo de futuras necesidades.

Las clases se describen en forma general como (ver Figura 4.4): La clase `ParametricQuery` abstrae parte del proceso de crear consultas paramétricas a una base de datos de secuencias, procurando armar la consulta a realizar a partir de archivos de configuración, la conexión con una base de datos, etc. `SeqTaxonomyEnricherDB` y `DesignDatabaseCreator` extienden `ParametricQuery` y escriben el resultado de sus consultas a archivos. `SequenceSplit` particiona un archivo `fasta` según tamaños definidos o el número de particiones que se requieran.

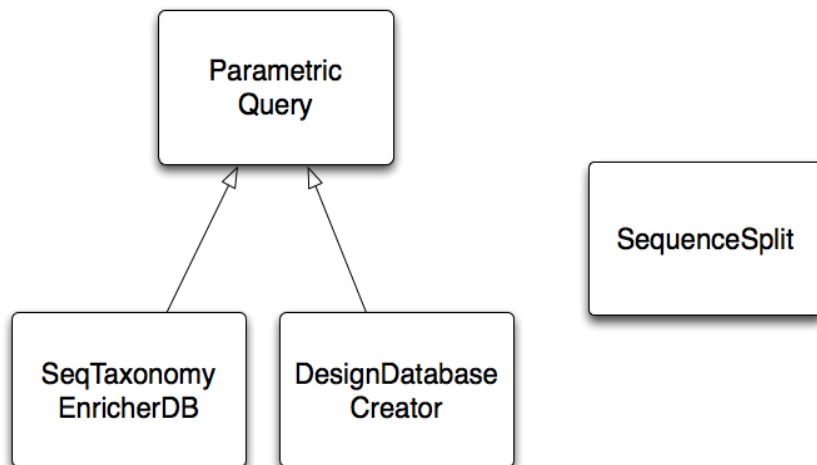


Figura 4.4: Diagrama de clases del módulo

Cuadro 4.10: Clase ParametricQuery

Nombre clase	ParametricQuery
Descripción	Guarda una consulta a una base de datos, utilizada para ser escrita a una base de datos en archivo ad-hoc por subclases.

Continúa en página siguiente...

Cuadro 4.10 – Continuación

Atributos	<ul style="list-style-type: none"> ▪ <code>dbName</code>: El nombre de la base de datos a usar. ▪ <code>queryTemplateFileName</code>: <i>Path</i> para el archivo que contiene la plantilla de la consulta a realizar. ▪ <code>queryVariableFileName</code>: <i>Path</i> para el archivo que contiene las variables de la plantilla de la consulta. ▪ <code>queryParamsFileName</code>: <i>Path</i> del archivo con los valores de las variables definidas en el archivo anterior. ▪ <code>outputFileName</code>: Nombre del archivo de salida. ▪ <code>dbCursor</code>: Un objeto <code>cursor</code> de la conexión con la base de datos. ▪ <code>query</code>: Un <code>string</code> con la consulta. ▪ <code>qpDict</code>: Un diccionario de uso interno de la clase.
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>: Inicializa una instancia de la clase, con los paths para todos los archivos requeridos. ▪ <code>start</code>: Construye la consulta y construye el objeto <code>dbCursor</code> para realizar dicha consulta.

Cuadro 4.11: Clase `SeqTaxonomyEnricherDB`

Nombre clase	<code>SeqTaxonomyEnricherDB</code>
Descripción	Extiende la clase <code>ParametricQuery</code> , definiendo el método <code>write</code> para realizar la escritura particular a archivo de la base de datos resultante de la consulta realizada.
Atributos	<ul style="list-style-type: none"> ▪ Los que hereda de la clase <code>ParametricQuery</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>write</code>: Escribe el resultado de la consulta a archivo.

Cuadro 4.12: Clase `SequenceSplit`

Nombre clase	<code>SequenceSplit</code>
Descripción	Particiona un archivo <code>fasta</code> en trozos más pequeños para posteriormente realizar cálculos distribuidos.

Continúa en página siguiente...

Cuadro 4.12 – Continuación

Atributos	<ul style="list-style-type: none"> ▪ fileInputName: El <i>path</i> al archivo <i>fasta</i>. ▪ partitionSize: El tamaño de la partición, o bien en número de secuencias por trozo o número de trozos. ▪ strategy: Estrategia para particionar: batch (particiones de tamaño partitionSize de secuencias) o parallel (número de particiones a realizar).
Métodos	<ul style="list-style-type: none"> ▪ __init__: Inicializa una instancia de la clase. ▪ batch_iterator: Función interna de la clase. ▪ split: Realiza la acción de partición.

Evaluación

`ParametricQuery` surgió como abstracción de los pasos en común que tienen `SeqTaxonomyEnricherDB` y `DesignDatabaseCreator`². Para el diseño posterior se requeriría realizar un objeto más que permita realizar lo que definiremos como una “base de datos de referencia” y “base de datos de simulación”. El comportamiento de estos objetos podría ser más relevante en el futuro al abstraer el proceso sistemático de mantener bases de datos de texto. La dependencia de la implementación está en la manera de conectar con la base de datos pero futuras modificaciones son acotadas a esta sección del sistema. A futuro se esperan realizar estas extensiones e incorporar los archivos de configuración usados en un sólo archivo.

`SequenceSplit` está en éste módulo por estar en el antiguo sistema en ésta sección, sin embargo se considera pertinente incluirlo en un futuro módulo de paralelización o llevarlo al módulo `ObjectsCollection`. Se creó como clase y no como funciones por motivos de claridad. En el esquema futuro de paralelización/*merge* en demanda este comportamiento será altamente demandado y su implementación es simple usando la herramienta de `BioPython`³.

Después de un rediseño, se hace necesario evaluar como se estructuran las herencias definidas en un contexto de preprocesamiento distribuido, tema que se espera abordar como trabajo futuro.

²La clase `DesignDatabaseCreator` no se describió por ser idéntica en su interfaz a `SeqTaxonomyEnricher`.

³Ver Capítulo de Implementación.

4.2.4. PrimerGenerators

Éste módulo contiene los generadores de *primers* a partir de los cuales se crean instancias de `PrimerSet`.

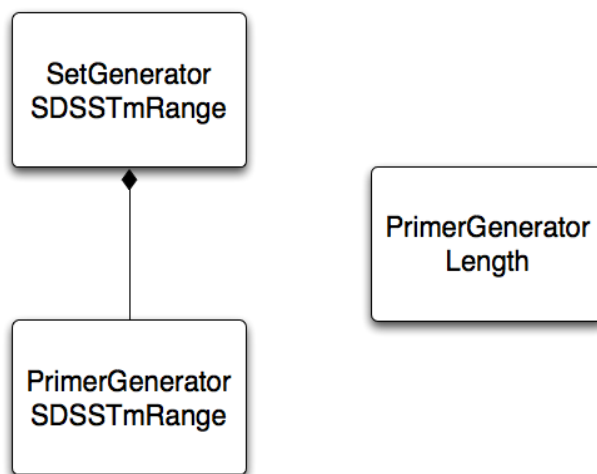


Figura 4.5: Diagrama de clases del módulo

El esquema general de las clases es (ver Figura 4.5): Las clases `PrimerGeneratorSDSSTmRange` y `PrimerGeneratorLength` toman un objeto `SeqRecords` de la librería `BioPython` y van generando *primers* de acuerdo a un criterio de región SDSS y un rango para la temperatura de *melting*, o de un rango de largo de los *primers*, respectivamente. `SetGeneratorSDSSTmRange` utiliza una instancia del objeto `PrimerGeneratorSDSSTmRange` para generar un `PrimerSet`.

Cuadro 4.13: Clase `PrimerGeneratorSDSSTmRange`

Nombre clase	<code>PrimerGeneratorSDSSTmRange</code>
Descripción	Un generador de secuencias de <i>primers</i> basándose en el cálculo de región SDSS propuesto por Miura et al.[19] y rango de temperatura de <i>melting</i> .

Continúa en página siguiente...

Cuadro 4.13 – Continuación

Atributos	<ul style="list-style-type: none"> ▪ sequences: Las secuencias a partir de las cuales obtener <i>substrings</i> para generar <i>primers</i>. Es un objeto <code>SeqRecords</code> de BioPython. ▪ tmMin: La temperatura de <i>melting</i> mínima requerida. ▪ tmMax: La temperatura de <i>melting</i> máxima permitida. ▪ santaLucia: Un objeto <code>SantaLucia</code>. ▪ input: Un objeto <code>InputValues</code>.
Métodos	<ul style="list-style-type: none"> ▪ __init__: Inicializa las instancias de la clase. Recibe los parámetros de temperatura en grados celsius y los transforma a grados Kelvin para manejo interno. ▪ __iter__: Permite crear un iterador sobre el objeto, mediante el uso de la directiva <code>yield</code> de Python, que convierte una función en un generador. ▪ __str__: Retorna una descripción del objeto.

Cuadro 4.14: Clase `SequenceSplit`

Nombre clase	<code>SequenceSplit</code>
Descripción	Utiliza un objeto <code>PrimerGeneratorSDSSTmRange</code> para generar un <code>PrimerSet</code> .
Atributos	<ul style="list-style-type: none"> ▪ filePathName: El <i>path</i> al archivo a partir del cual obtener <i>primers</i>, en formato <i>fasta</i>. ▪ input: Un objeto <code>InputValues</code>.
Métodos	<ul style="list-style-type: none"> ▪ __init__: Inicializa las instancias de la clase. ▪ __call__: Construye el <code>PrimerSet</code>. Los atributos iniciales para los <i>primers</i> son: <code>Name</code>, <code>Sequence</code>, <code>Gen_SDSS_F</code> (umbral con que el parámetro <i>F</i> del cálculo de Miura fue superado), <code>Gen_SDSSRegion</code> (el largo de la región SDSS), <code>Gen_Coord_Orig</code>, <code>Orientation</code> (<i>forward</i> = 'F' o <i>reverse</i> = 'R'). ▪ __str__: Retorna una descripción del objeto.

Evaluación

La separación del `PrimerGeneratorSDSSTmRange` y `SetGeneratorSDSSTmRange` van en la dirección de poder abstraer el algoritmo con que se generan *primers* de la creación del conjunto mismo. No se incluye en la descripción detallada el objeto `PrimerGeneratorLength` por tener una interfaz análoga a la de `PrimerGeneratorSDSSTmRange`, salvo que los atributos iniciales con que parten los *primers* son distintos. Es necesario contar con algún algoritmo para generar *primers* y no considerar todas las subsecuencias posibles pues de ser así la cantidad de secuencias con las cuales trabajar serían demasiadas en contraste con las pocas que podrían quedar después de cualquier filtro básico. Se evaluó incorporar los filtros que se verán en la subsección siguiente como métodos que permiten a estos objetos filtrar la generación de *primers* pero se descartó por infactible dada la arquitectura actual del sistema y algunas pruebas de concepto básicas. El comportamiento de estas clases solo es requerido al momento de generar los *primers* pero por motivos de claridad, en función de la arquitectura general descrita se consideró valioso mantenerlos en un módulo aparte.

Pese a que pareciera que no hay muchos criterios adicionales más, estos si pueden variar enormemente dependiendo del alcance de las extensiones que se quieran realizar, por ejemplo, si quisieramos incorporar *primers* exitósos en PCR's reales como valores de entrada.

4.2.5. PrimerEnhancers

Éste es el módulo central del trabajo llevado a cabo en esta memoria. Todas las operaciones que se aplican a los primers se conceptualizaron como **Enhancers**, que toman un `PrimerSet`, realizan operaciones con el y lo retornan con más atributos, menos *primers*, *metadata*, entre otras.

El esquema general de las clases es (ver Figura 4.6): La clase `FilterCutValue` es un filtro en su forma más abstracta; toma un atributo de la matriz de *primers* y verifica que sea mayor o igual, o menor o igual, que un valor de umbral dado, dejando los *primers* en un `PrimerSet` que satisfagan la condición descrita. La clase `AttributeAppend` es la clase abstracta con la cual se añaden atributos a un `PrimerSet`, y todas las otras subclases que heredan de ella solo implementan la función explícita que les diferencia. El resto de las clases con el sufijo `FileAppend` en sus nombres son como su nombre lo indica, clases que toman un `PrimerSet` y le asocian un archivo (en general alguna base de datos de texto). La clase `Backup` es un **Enhancer** que respalda en archivos el estado del `PrimerSet`.

Cuadro 4.15: Clase `FilterCutValue`

Nombre clase	<code>FilterCutValue</code>
Descripción	Clase para filtrado de <i>primers</i> de acuerdo a algún valor umbral.

Continúa en página siguiente...

Cuadro 4.15 – Continuación

Atributos	<ul style="list-style-type: none"> ▪ name: El nombre del filtro. ▪ type: El tipo de filtro, definido con valores constantes. Los valores posibles son ‘Threshold’, ‘Percentage’ y ‘Ranking’. ▪ threshold: El valor umbral para el tipo de filtro definido. ▪ parameter: El nombre del atributo para realizar la comparación contra el threshold y decidir si un <i>primer</i> continúa en el <code>PrimerSet</code> o no. ▪ cutFromAbove: Valor booleano para decidir si el valor de parameter para un <i>primer</i> dado debe ser menor o igual (<code>True</code>), o mayor o igual (<code>False</code>) que threshold, por defecto es <code>False</code>.
Métodos	<ul style="list-style-type: none"> ▪ __init__: Inicializa los valores requeridos para el filtro. ▪ __call__(primerSet): Realiza el filtrado de los <i>primers</i>. Cuando el tipo de umbral es ‘Threshold’, se verifica que el atributo de un <i>primer</i> supere o iguale el umbral para pasar el filtro. En los otros dos casos, se toma el vector columna del atributo correspondiente, se ordena y calcula el valor del atributo en la posición que determine threshold según el tipo de umbral definido. Así, se utiliza ese nuevo umbral como valor de corte. Estos umbrales son distintos para <i>primers forwards</i> y <i>primers reverse</i>. ▪ __str__: Retorna una descripción del filtro, incluyendo información de la cantidad de secuencias que habian antes de filtrar, la cantidad filtrada y las que quedan, junto a un porcentaje de filtrado.

Cuadro 4.16: Clase `AttributeAppend`

Nombre clase	<code>AttributeAppend</code>
Descripción	La clase “abstracta” para agregar atributos a un <code>PrimerSet</code> .

Continúa en página siguiente...

Cuadro 4.16 – Continuación

Atributos	<ul style="list-style-type: none"> ▪ attributeList: Una lista con los nombres de los atributos a añadir. ▪ parameterList: Una lista con los nombres de los parámetros que considerar de un <i>primer</i> para calcular los atributos a añadir. ▪ name: Un nombre para el Enhancer ▪ description: Una descripción del filtro opcional, por defecto es vacía.
Métodos	<ul style="list-style-type: none"> ▪ __init__(attributeList,parameterList,name): Inicializa los valores requeridos. ▪ __call__(primerSet): Añade los atributos definidos en attributeList en función de los parámetros de parameterList. En función de la porción de la matriz de <i>primers</i> y los atributos input y filesDict del argumento, invoca a myFunction para obtener el nuevo conjunto de secuencias que corresponde. ▪ __str__: Retorna una descripción. ▪ myFunction(x,input,filesDict): Retorna un conjunto de atributos para las secuencias a partir de x, utilizando datos de input y filesDict. En ésta clase en particular es un método abstracto.

Cuadro 4.17: Clase LengthAppend

Nombre clase	LengthAppend
Descripción	Una clase AttributeAppend que añade el atributo del largo de las secuencias de los <i>primers</i> .
Atributos	<ul style="list-style-type: none"> ▪ Los que hereda de AttributeAppend.
Métodos	<ul style="list-style-type: none"> ▪ myFunction(x,input,filesDict): Recibe las secuencias, retorna su largo. ▪ Los que hereda de AttributeAppend.

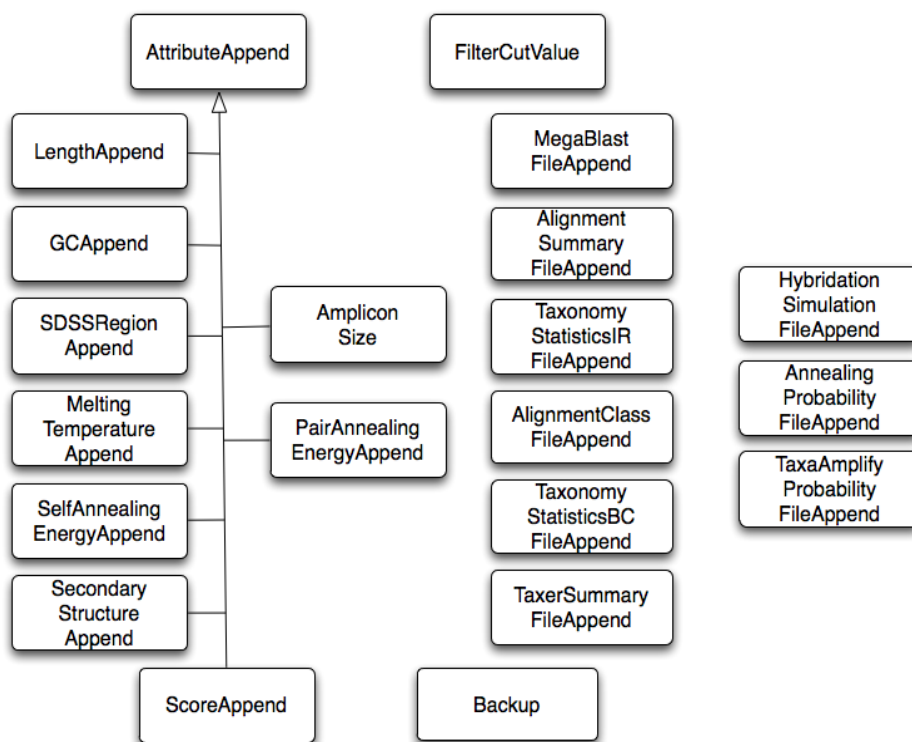


Figura 4.6: Diagrama de clases del módulo

Cuadro 4.18: Clase GCAppend

Nombre clase	GCAppend
Descripción	Añade el contenido <i>GC</i> de las secuencias de <i>primers</i> (porcentaje de letras <i>G</i> y <i>C</i> del total).
Atributos	<ul style="list-style-type: none"> ▪ Los que hereda de <code>AttributeAppend</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>myFunction(x, input, filesDict)</code>: Recibe las secuencias, retorna su contenido <i>GC</i>. ▪ Los que hereda de <code>AttributeAppend</code>.

Cuadro 4.19: Clase SelfAnnealingEnergyAppend

Nombre clase	SelfAnnealingEnergyAppend
Descripción	Añade un atributo con la energía libre de Gibbs del <i>primer</i> hibridando con copias de sí mismo. Utiliza para ello el programa <code>hybrid-min</code> del software Unafold.
Atributos	<ul style="list-style-type: none"> ▪ <code>prefix</code>: Un prefijo para los archivos de <i>input</i> y <i>output</i> al utilizar <code>hybrid-min</code>. Por defecto es vacío. ▪ Los que hereda de <code>AttributeAppend</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>myFunction(x,input,filesDict)</code>: Prepara el archivo de entrada para utilizar <code>hybrid-min</code>, lo invoca y posteriormente lee la salida para retornar el vector de atributos con las energías libres. ▪ Los que hereda de <code>AttributeAppend</code>.

Cuadro 4.20: Clase SecondaryStructureAppend

Nombre clase	SecondaryStructureAppend
Descripción	Añade un atributo con la energía libre de Gibbs del <i>primer</i> hibridando con sí mismo. Utiliza para ello el programa <code>hybrid-ss-min</code> del software Unafold.
Atributos	<ul style="list-style-type: none"> ▪ <code>prefix</code>: Un prefijo para los archivos de <i>input</i> y <i>output</i> al utilizar <code>hybrid-ss-min</code>. Por defecto es vacío. ▪ Los que hereda de <code>AttributeAppend</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>myFunction(x,input,filesDict)</code>: Prepara el archivo de entrada para utilizar <code>hybrid-ss-min</code>, lo invoca y posteriormente lee la salida para retornar el vector de atributos con las energías libres. ▪ Los que hereda de <code>AttributeAppend</code>.

Cuadro 4.21: Clase ScoreAppend

Nombre clase	ScoreAppend
Descripción	Añade un puntaje en función de un atributo dado, a partir de un rango para dicho atributo y opcionalmente un valor óptimo.

Continúa en página siguiente...

Cuadro 4.21 – Continuación

Atributos	<ul style="list-style-type: none"> ▪ <code>score</code>: Un objeto <code>TriangleScore</code>. ▪ Los que hereda de <code>AttributeAppend</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>myFunction</code>: Recibe un atributo, retorna el puntaje para dicho atributo. ▪ Los que hereda de <code>AttributeAppend</code>.

Cuadro 4.22: Clase `AmpliconSize`

Nombre clase	<code>AmpliconSize</code>
Descripción	Opera sobre un <code>PrimerPairSet</code> . Añade atributos de posición inicial, posición final y largo de la región de amplificación de una pareja de <i>primers</i> en las coordenadas de la referencia.
Atributos	<ul style="list-style-type: none"> ▪ Los que hereda de <code>AttributeAppend</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>myFunction</code>: Recibe para <i>forward</i> y <i>reverse</i> la secuencia del <i>primer</i>, su secuencia de origen y posición de origen. ▪ Los que hereda de <code>AttributeAppend</code>.

Cuadro 4.23: Clase `PairAnnealingEnergyAppend`

Nombre clase	<code>PairAnnealingEnergyAppend</code>
Descripción	Opera sobre un <code>PrimerPairSet</code> . Añade la energía libre de Gibbs de la pareja de <i>primers</i> hibridando entre sí usando <code>hybrid-min</code> .
Atributos	<ul style="list-style-type: none"> ▪ Los que hereda de <code>AttributeAppend</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>myFunction</code>: Calcula la energía. ▪ Los que hereda de <code>AttributeAppend</code>.

A continuación se exponen los `Enhancers` que añaden archivos. Todos tienen una estructura similar: además de los parámetros que les son necesarios para calcular lo que deben añadir, reciben opcionalmente un prefijo para añadir a los nombres de los archivos utilizados como entrada y salida, y definen los métodos `__init__`, que inicializa las instancias de la

clase, `__call__(primerSet,outputFilekey)` que realiza la labor requerida, dejando el resultado en un archivo cuya ruta queda en el atributo `filesDict` del `primerSet` con la llave `outputFileKey`, y `__str__` que retorna descripciones de cada `Enhancer`.

Cuadro 4.24: Clase `MegaBlastFileAppend`

Nombre clase	<code>MegaBlastFileAppend</code>
Descripción	Toma el <code>PrimerSet</code> y realiza un <code>megablast</code> contra una base de datos de referencia de secuencias previamente formateadas para ello.
Atributos	<ul style="list-style-type: none"> ▪ <code>referenceDB</code>: <i>Path</i> del archivo <code>fasta</code> con las secuencias de referencia formateadas. ▪ <code>prefix</code>: Un prefijo para usar en los nombres de los archivos de entrada y salida. ▪ <code>sdssFlag</code>: Un valor booleano para saber si usar la secuencia completa o sólo su región SDSS en el alineamiento. Valor por defecto: <code>False</code>. ▪ <code>sdssName</code>: El nombre del atributo que contiene la región SDSS (si el atributo anterior de esta clase tiene valor <code>True</code>).
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>, <code>__call__</code> y <code>__str__</code>.

Cuadro 4.25: Clase `AlignmentSummaryFileAppend`

Nombre clase	<code>AlignmentSummaryFileAppend</code>
Descripción	Toma una salida de <code>megablast</code> y calcula posiciones de alineamiento típicas respecto a un organismo de referencia (vía “cambios de coordenadas”). También guarda en un archivo separado el listado de secuencias que alinearon contra un <i>primer</i> dado, para cada uno.
Atributos	<ul style="list-style-type: none"> ▪ <code>blastFile</code>: <i>Path</i> del archivo con la salida de <code>megablast</code>. ▪ <code>coordinator</code>: Un objeto <code>Coordinator</code>. ▪ <code>prefix</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>, <code>__call__</code> y <code>__str__</code>.

Cuadro 4.26: Clase `AlignmentFileClassAppend`

Nombre clase	<code>AlignmentClassAppend</code>
---------------------	-----------------------------------

Continúa en página siguiente...

Cuadro 4.26 – Continuación

Descripción	<p>Clasificador de pares de <i>primers</i> y secuencias de acuerdo al siguiente esquema:</p> <ul style="list-style-type: none"> ▪ ‘Y’: Si la secuencia contiene estrictamente la región de alineamiento típica del <i>primer</i> en las coordenadas de referencia y alinean según <i>megablast</i>. ▪ ‘N’: Si la secuencia contiene estrictamente la región de alineamiento típica del <i>primer</i> en las coordenadas de referencia y no alinean según <i>megablast</i>. ▪ ‘U’: Si la secuencia y la región de alineamiento típica del <i>primer</i> intersectan nulo y no alinean según <i>megablast</i>. ▪ ‘X’: El resto de los casos.
Atributos	<ul style="list-style-type: none"> ▪ <code>coordsRefFile</code>: <i>Path</i> del archivo con las coordenadas de referencia para las secuencias de la base de datos de referencia, con la primera y última coordenada para cada una. ▪ <code>sequencesListFile</code>: <i>Path</i> del archivo con los <i>primers</i> y las secuencias que alinean con cada uno. ▪ <code>alignmentStatsFile</code>: <i>Path</i> del archivo con las estadísticas de alineamiento de acuerdo a un organismo de referencia. ▪ <code>prefix</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>, <code>__call__</code> y <code>__str__</code>.

Cuadro 4.27: Clase `TaxerSummaryFileAppend`

Nombre clase	<code>TaxerSummaryFileAppend</code>
Descripción	Toma una clasificación de pares de <i>primers</i> -secuencias y resume los valores por categoría de clasificación según taxonomía.
Atributos	<ul style="list-style-type: none"> ▪ <code>classFile</code>: <i>Path</i> del archivo con la clasificación. ▪ <code>seqsTaxFile</code>: <i>Path</i> del archivo con las secuencias enriquecidas taxonómicamente. ▪ <code>prefix</code>.

Continúa en página siguiente...

Cuadro 4.27 – Continuación

Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>, <code>__call__</code> y <code>__str__</code>.
----------------	--

Cuadro 4.28: Clase `TaxonomyStatisticsIRFileAppend`

Nombre clase	<code>TaxonomyStatisticsIRFileAppend</code>
Descripción	<p>Añade atributos y un archivo a partir de un archivo, con varios estadísticos tipo <i>Information Retrieval</i> del comportamiento de los <i>primers</i> en cada taxa.</p> <p>Particiona el espacio taxonómico en tres categorías: taxas objetivo, no objetivo y el complemento. Para cada una de ellas, se calculan <i>precision</i>, <i>recall</i>, $precision \times recall$, $F_{measure}(precision, recall)$ y número de organismos. Se apoya en una clasificación en cuatro estados.</p>
Atributos	<ul style="list-style-type: none"> ▪ <code>taxaFile</code>: <i>Path</i> del archivo con las clasificaciones por taxa. ▪ <code>prefix</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>, <code>__call__</code> y <code>__str__</code>.

Cuadro 4.29: Clase `HybridationSimulationFileAppend`

Nombre clase	<code>HybridationSimulation</code>
Descripción	Opera sobre un <code>PrimerPairSet</code> . Agrega un archivo con simulación de hibridación de las secuencias de los <i>primers</i> contra una base de datos de secuencias para simulación.
Atributos	<ul style="list-style-type: none"> ▪ <code>simulationSeqs</code>: <i>Path</i> del archivo con las secuencias a usar para simular, en formato <code>fasta</code>. ▪ <code>prefix</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>, <code>__call__</code> y <code>__str__</code>.

Cuadro 4.30: Clase `AnnealingProbabilityFileAppend`

Nombre clase	<code>AnnealingProbabilityFileAppend</code>
Descripción	Opera sobre un <code>PrimerPairSet</code> . Calcula la probabilidad de un <i>primer</i> de amplificar una secuencia en una posición de ella.

Continúa en página siguiente...

Cuadro 4.30 – Continuación

Atributos	<ul style="list-style-type: none"> ▪ <code>hybridMinOut</code>: <i>Path</i> del archivo con el resultado de simulación vía <code>hybrid-min</code>. ▪ <code>prefix</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>, <code>__call__</code> y <code>__str__</code>.

Cuadro 4.31: Clase `TaxaAmplifyProbabilityFileAppend`

Nombre clase	<code>TaxaAmplifyProbabilityFileAppend</code>
Descripción	Calcula la probabilidad de amplificar una taxa dada las probabilidades de amplificación por hibridación.
Atributos	<ul style="list-style-type: none"> ▪ <code>probFile</code>: <i>Path</i> del archivo con el resultado del calculo de probabilidad para pares <i>primer-secuencia</i>. ▪ <code>prefix</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>, <code>__call__</code> y <code>__str__</code>.

Evaluación

El nombre de `Enhancers` es dado que todos comparten el comportamiento de tomar un `PrimerSet` y modificarlo de algún modo. El colocarlos todos juntos en un sólo módulo permite mantener la conceptualización presentada en la arquitectura del sistema. Como se mencionó en el módulo `ObjectsCollection`, `Backup` se considera incorporarlo al objeto `PrimerSet` como función.

La abstracción realizada en el objeto `AttributeAppend` permite realizar todas las operaciones de iterar sobre la matriz de *primers* y realizar las operaciones necesarias que garantizan el comportamiento esperado, independiente de las funciones puntuales que permiten calcular los atributos a añadir. Sin embargo, una abstracción similar para los objetos con sufijo `FileAppend` no fue posible para el alcance de esta memoria, por razones de tiempo y que dado lo implementado, dicha abstracción sólo sería una declaración de interfaz, pero en el trabajo futuro que se expondrá, dicha abstracción será necesaria considerando los diversos entornos operativos que el sistema pudiese tener, sobre todo en un ambiente de trabajo paralelo o distribuido. La distinción entre tener `Enhancers` que añaden atributos o añaden archivos está en que los que hacen lo primero, sólo lo hacen para cada *primer*, en cambio si se requiere asociar a cada *primer* un conjunto de secuencias y algún valor de alineamiento o hibridación

para cada combinación, manejar en la memoria de la instancia todos esos datos se puede tornar inmanejable, y por ello se opta por ir asociando bases de datos de texto que se puedan ir enlazando como *input* y *output*.

En el diseño detallado no se incluyeron las clases `MeltingTemperatureAppend` ni `SDSS-RegionAppend` por ser similares en su comportamiento a lo descrito en el módulo `Primers-Generators`, con la diferencia que en este caso añaden los atributos que en la clase `PrimerGeneratorSDSSTmRange` se incluyen desde el comienzo en el `PrimerSet`. Tampoco se incluyó la clase `TaxonomyStatisticsBCFileAppend` por ser análoga a `TaxonomyStatistics-IRFileAppend`.

El valor de tener los **Enhancers** definidos en este esquema es que permiten hacer que en gran medida, estos sean **altamente parametrizables e intercambiables entre sí**, proporcionando una infinidad de opciones para diseñar estrategias específicas para distintos problemas. Las especificaciones son lo bastante precisas y acotadas para hacer que sean comportamientos factibles y no dependientes de las implementaciones que posean. Sin embargo, se evalúa la necesidad de abstraer los **Enhancers** que añaden archivos, para mantener aun más flexible y simple el esquema propuesto y que permita realizar a futuro la posibilidad de portar el sistema a distintos entornos de trabajo.

En un comienzo, los objetos `PrimerSet` y `PrimerPairSet` se concibieron de forma separada, requiriendo objetos y **Enhancers** separados. Sin embargo, avanzando en la implementación se consideró que son el segundo una extensión del primero, y dadas las interfaces de los **Enhancers**, no era necesario hacer tal distinción.

4.2.6. DesignStrategies

En este módulo se encuentran todas las clases que implementan estrategias de diseño. De esta librería los diseñadores de *primers* pueden eventualmente escoger las estrategias con las cuales diseñarán *primers*, y los diseñadores de estrategias extenderán, agregando más clases. Para efectos de este trabajo de memoria, sólo hay una clase: `LBMGDesignStrategy` a partir de la cual se puede abstraer una interfaz que todas las otras estrategias de diseño deben implementar:

Cuadro 4.32: Clase `LBMGDesignStrategy`

Nombre clase	<code>LBMGDesignStrategy</code>
Descripción	Estrategia de diseño existente en el LBMG para el diseño de <i>primers</i> para metagenómica.

Continúa en página siguiente...

Cuadro 4.32 – Continuación

Atributos	<ul style="list-style-type: none"> ▪ <code>primerGenerator</code>: Una instancia de un objeto de tipo <code>SetGenerator</code> para generar <i>primers</i>. ▪ <code>primerSet</code>: Una instancia de <code>PrimerSet</code>. ▪ <code>primerPairs</code>: Una instancia de <code>PrimerPairSet</code>.
Métodos	<ul style="list-style-type: none"> ▪ <code>__init__</code>: Inicializa los atributos de una clase. ▪ <code>__str__</code>: Retorna una descripción de la estrategia. ▪ <code>recover</code>: Realiza un <code>recover</code> del <code>primerSet</code> o <code>primerPairs</code> de la clase. ▪ <code>createPairs</code>: Define el atributo <code>primerPairs</code> a partir del atributo <code>primerSet</code>. ▪ <code>showPrimers</code>: Muestra una descripción del <code>primerSet</code> de la clase. ▪ <code>showPairs</code>: Muestra una descripción del <code>primerPairs</code> de la clase. ▪ <code>filterPrimers</code>: Aplica los <code>Enhancers</code> definidos para el <code>primerSet</code> de la clase. ▪ <code>filterPairs</code>: Aplica los <code>Enhancers</code> definidos para el <code>primerPairs</code> de la clase.

Evaluación

El objetivo de estas clases es servir de contenedoras para estrategias que pueden intercambiarse a futuro. En cierto modo son una versión más flexible del patrón de diseño **Strategy**, y dada la conceptualización del diseño de *primers* con los objetos `PrimerSet` y `PrimerPairSet` junto a los `Enhancers`, la clase diseñada es sólo la primera de muchas estrategias de diseño de *primers*. Manteniendo la arquitectura del diseño del resto de las clases, no hay dependencias limitantes en la implementación.

El diseño de esta clase fue inspirado en parte entre los patrones **Template**, **Strategy** y **Facade**, pero sin definir las clases adecuadas en esta primera etapa. Sin embargo, las estrategias que se diseñen puede definir una interfaz adecuada para brindar extensibilidad a la estrategia mediante el patrón **Template**, se pueden crear diversos contenedores según emer-

jan de futuras aplicaciones utilizando **Facade** y brindar una interfaz definida en **Strategy** para las partes que necesiten cambios a futuro en los algoritmos.

Capítulo 5

Implementación del Sistema

En éste capítulo se abordarán aspectos relacionados con la implementación del sistema. Se describirá el entorno de trabajo y la arquitectura física del desarrollo, y para cada módulo, se detallará para las clases y funciones que contiene dicho módulo aspectos relevantes de la implementación, formalizaciones de las fórmulas matemáticas usadas, entre otras consideraciones de relevancia para la comprensión de los aspectos de la complejidad del sistema que sustentarán el diseño y extensión del mismo a un entorno de producción.

5.1. Entorno de Trabajo

Cómo se explicó en la sección pasada, el lenguaje de programación escogido fue Python. A continuación, un listado de los programas y herramientas utilizados:

Cuadro 5.1: Software utilizado para el desarrollo.

Nombre	Descripción
SVN	Herramienta para versionar código fuente. http://subversion.tigris.org/
Eclipse	IDE para desarrollo, integra con SVN. http://www.eclipse.org/
PyDec	<i>Plug-in</i> para Eclipse para desarrollar en Python. http://pydev.org/

Continúa en página siguiente. . .

Cuadro 5.1 – Continuación

MySQL	Motor y Sistema de administración de bases de datos. http://www.mysql.com/
EMBOSS	Conjunto de herramientas bioinformáticas. En particular el uso de la herramienta de alineamiento múltiple <code>needle</code> . http://emboss.sourceforge.net/
Blast	Conjunto de herramientas bioinformáticas. Para uso de <code>megablast</code> y <code>formatdb</code> . ftp://ftp.ncbi.nlm.nih.gov/blast/
Unafold	Software para cálculos termodinámicos de hibridaciones de secuencias. Uso de programas <code>hybrid-ss-min</code> e <code>hybrid-min</code> . http://dinamelt.bioinfo.rpi.edu/download.php
Awk	Software de amplio uso para tratamiento de archivos de texto, usado en una parte específica del sistema y utilizado para ilustrar el uso de herramientas externas. http://cm.bell-labs.com/cm/cs/awkbook/index.html
BioPython	Librería de objetos y métodos para bioinformática en Python. http://biopython.org

5.2. Implementación de Clases

5.2.1. PrimerConstants

En éste módulo todas las constantes se definen con los tipos básicos de Python: `int`, `float`, `string`, `list` y `dict`. Algunas constantes que definen “estados” de algunos objetos se incluyen también. A futuro, algunas opciones de otros objetos y futuras extensiones deberán incluir más constantes acá.

5.2.2. ObjectsCollection

- Las clases `ThermoDynamicParameters` y `SantaLucia` son esencialmente contenedores que definen los atributos mínimos que se requieren para que otras tareas se puedan

efectivamente realizar.

- **InputValues** incorpora una verificación extensiva de los parámetros haciendo *cast* de **string** a los tipos de datos necesarios (**int**, **float**, etc.) para cuando se recuperan los valores desde un archivo.
- **TriangleScore** implementa un puntaje de forma triangular, que es una instancia particular de una función unimodal no negativa, es decir, cuyo valor mínimo es mayor o igual a cero y tiene sólo un punto máximo, pudiendo tener diversas formas en los dos subintervalos definidos y haciéndose cargo de diversos casos de borde. Se incorpora en la implementación con fines ilustrativos un **QuadraticScore** que es esencialmente igual a **TriangleScore** pero cuadrático.
- **PrimerSet** tiene varios métodos implementados que procuran crear en formatos “legibles por humanos” los datos que contengan los atributos del objeto. La función **recover** está ligada a esas otras funciones pues a partir del formato que ellas definen, recupera los valores desde archivos de texto. Este aspecto es potencialmente abstraible a futuro considerando la complejidad futura del objeto, sobre todo si se desea particionar los conjuntos de datos para distribuir el cálculo y también para serializar algunas operaciones.
- **PrimerPairSet** tiene principalmente un método de creación de pares de *primers* a partir de un **PrimerSet**, juntando a cada *primer forward* con cada *primer reverse*. Se considera a futuro incorporar otros esquemas que filtre el número de parejas, como se hace en la etapa de generación de *primers*.
- **CoordinatesCalculator**, **FileCoordinator** y **Coordinator** se implementan con una fórmula sencilla usando expresiones regulares.

5.2.3. Preprocess

- **ParametricQuery** requiere importar el módulo **MySQLdb**, y las subclases tienen implementaciones directas según las directrices necesarias para construir las bases de datos de texto.
- **SequenceSplit**: Particiona un conjunto de secuencias usando **BioPython**.

En el módulo se implementa también las funciones que se esperan realizar en el preprocesamiento. En particular para lo implementado en este trabajo de memoria, se crean las bases de datos de texto correspondientes, se utiliza el programa **formatdb** para crear bases de datos con las que trabaja **blast** y realizar alineamiento de secuencias, se alinean secuencias con el programa **needle** del cual se obtienen los archivos de alineamiento en una línea y de coordenadas para las secuencias relevantes.

5.2.4. PrimerGenerators

- Como se discutió en la sección correspondiente de diseño, se usó la directiva `yield` de Python para permitir definir generadores a partir de código que se escribe como una función.
- En la clase `PrimerGeneratorSDSSTmRange` se recorre desde un extremo de la secuencia dada hacia atrás, hasta obtener una secuencia con una región SDSS cuyo valor F supere un umbral dado. Por definición de la región SDSS, esta es menor (en largo) a la que se requeriría para obtener *primers* con temperaturas de *melting* en rangos aceptables comunes para PCRs (60, 70 grados celsius). Internamente las temperaturas se manejan en grados kelvin. Obtenidos todos los posibles *primers* que parten en una posición dada de la secuencia de origen, el origen se desplaza, hasta recorrer toda la secuencia, y a su vez hasta iterar sobre todas las secuencias recibidas. En caso de estar generando *primers reverse*, se calcula previamente el reverso complementario de la secuencia de origen.

5.2.5. PrimerEnhancers

- En un comienzo `PrimerSet` contenía dos conjuntos de *primers*: los *forward* y los *reverse*. Por ello, `FilterCutValue` no hacía distinciones para calcular los umbrales de corte respectivos. Sin embargo, al decidir después mantener sólo un conjunto de *primers*, se necesitó que los umbrales de corte relativo cuando se necesita un porcentaje o un ranking de *primers* fuesen distinto, pues a la etapa de la formación de parejas se debe llegar idealmente con ambos conjuntos de *primers* no vacíos, y por razones estructurales los valores umbrales para los atributos de ambos conjuntos puede diferir ampliamente.
- Todos los `Enhancers` que añaden archivos, o atributos usando programas externos, utilizan archivos de entrada y salida que quedan en directorios del entorno de trabajo pero que están escritos como valores constantes en distintas partes del código. Se considera necesario definir una interfaz adecuada que permita portar y configurar esto de forma adecuada.
- El archivo generado por `AlignmentSummaryFileAppend` escribe en un archivo separado por tabs:
 - `PrimerName`: El nombre del *primer*.
 - `MinStartPos`: Posición mínima de inicio de alineamiento.
 - `MaxStartPos`: Posición máxima de inicio de alineamiento.
 - `MinEndPos`: Posición mínima de fin de alineamiento.
 - `MaxEndPos`: Posición máxima de fin de alineamiento.
 - `StartPosAverage`: Media de posiciones de inicio.

- **EndPosAverage**: Media de posiciones de término.
 - **StartPosMedian**: Mediana de posiciones de inicio.
 - **EndPosMedian**: Mediana de posiciones de término.
 - **Hits**: Número de alineamientos.
- **AlignmentClassFileAppend** clasifica según un esquema que o bien podría considerarse como un esquema de clasificación binaria, posibilitando el cálculo de las medidas de **Sensitivity** y **Specificity**, o bien medidas inspiradas en el área de Recuperación de la Información, como **Precision** y **Recall**. Si consideramos que en las categorías definidas:
- ‘**Y**’: Si la secuencia contiene estrictamente la región de alineamiento típica del *primer* en las coordenadas de referencia **y** alinean según **megablast**.
 - ‘**N**’: Si la secuencia contiene estrictamente la región de alineamiento típica del *primer* en las coordenadas de referencia **y no** alinean según **megablast**.
 - ‘**U**’: Si la secuencia y la región de alineamiento típica del *primer* intersectan nulo **y no** alinean según **megablast**.
 - ‘**X**’: El resto de los casos.

consideramos a ‘**Y**’ como *True positive* (T_p), ‘**N**’ como *False positive* (F_p), ‘**U**’ como *True negative* (T_n) y ‘**X**’ como *False Negative* (F_n), entonces:

$$Sensitivity = \frac{Y}{Y + X} \quad Specificity = \frac{U}{U + N}$$

donde estos valores se calculan para cada primer según los organismos presentes en las tres categorías posibles de taxa (objetivo, no objetivo y complemento).

Por otro lado, utilizando un esquema de recuperación de la Información, podemos definir **Precision** y **Recall** como:

$$Precision = \frac{\# \text{ secuencias cuya región en la referencia intersecta no vacío con la región del } primer \text{ y alinean con él en la categoría de taxa}}{\# \text{ secuencias cuya región en la referencia intersecta no vacío con la región del } primer \text{ y alinean con él en todas las taxa}}$$

$$Recall = \frac{\# \text{ secuencias cuya región en la referencia intersecta no vacío con la región del } primer \text{ y alinean con él en la categoría de taxa}}{\# \text{ secuencias cuya región en la referencia intersecta no vacío con la región del } primer \text{ o alinean con él en la categoría de taxa}}$$

Estas dos maneras de calcular los estadísticos son llevadas a cabo por `TaxonomyStatisticsBCFileAppend` y `TaxonomyStatisticsIRFileAppend`, respectivamente, y son con los archivos que estos `Enhancers` producen a partir del cual se pueden filtrar *primers* para armar las parejas.

- `AmpliconSize` calcula las posiciones de los *primers* respecto al organismo de referencia con las siguientes fórmulas:

$$\text{PosiciónReferencia}(p_f) = (\text{Origen}(p_f) - \text{largo}(p_f))_{\text{Referencia}}$$

$$\text{PosiciónReferencia}(p_r) = (\text{largo}(\text{SecuenciaOrigen}(p_r)) - \text{Origen}(p_f) + \text{largo}(p_r))_{\text{Referencia}}$$

$$\text{Largo Amplicon} = \text{PosiciónReferencia}(p_f) - \text{PosiciónReferencia}(p_r)$$

- `HybridationSimulation` calcula mediante `hybrid-min` la energía libre de Gibbs (ΔG) de la última letra de un *primer* hibridando en cada letra de las secuencias de simulación, para todos los *primers* y para todas las secuencias de simulación. Posteriormente, estas energías se mapean a probabilidad de hibridación mediante la fórmula:

$$\mathbb{P}(\text{primer } p \text{ hibride en la letra } i \text{ de la secuencia } s) = \frac{e^{-\frac{\Delta G_i}{RT}}}{F_p}$$

con $F_p = \sum_{i \in s} e^{-\frac{\Delta G_i}{RT}}$, R la constante de los gases y T la temperatura de operación del PCR.

- La información por taxas se condensa calculando la probabilidad de hibridación de la pareja contra una secuencia dada a un nivel mayor que un umbral dado, indicando el tamaño del amplicon que forman, estimando una probabilidad de amplificación conjunta (suponiendo hibridaciones independientes) y calculando la cantidad de secuencias de las taxas objetivo y no objetivo son cubiertas (en sentido de hibridación satisfactoria) por la pareja con alta probabilidad.

5.2.6. DesignStrategies

La clase `LBMGDesignStrategy` es la que implementó mediante una cadena de `Enhancers` específicos, el poder portar el antiguo *pipeline* de diseño de *primers* al sistema desarrollado en esta memoria.

Los pasos son:

- Se crean los *primers* con el criterio SDSS y se calcula a la vez la temperatura de *melting*.
- Se filtran por contenido GC (primero añadiendo el atributo, luego filtrando por él).

- Se filtran por largo de secuencias de acuerdo a un puntaje
- Se filtran por energía de estructuras secundarias.
- Se filtran por energía de hibridaciones con copias de sí mismos.
- Se realiza un blast de las secuencias que quedan, considerando solo su región SDSS.
- Se comparan posiciones de alineamiento de acuerdo a un sistema de coordenadas definido mediante alineamiento múltiple con un organismo de referencia, en el caso del LBMG, para diseño de *primers* en metagenómica para organismos biomineros ese organismo es *Escherichia Coli*.
- Se clasifican los pares *primers*-secuencias.
- Se condensan valores por taxa.
- Se calculan atributos de ***Precision*** y ***Recall*** por categorías taxonómicas y se filtran los primers de acuerdo a ello.
- Se forman parejas.
- Se filtra por largo esperado de amplicón en función del organismo de referencia.
- Se filtran los que tengan una energía libre de Gibbs de hibridación entre *primer forward* y *primer reverse* demasiado alta.
- Se simula la hibridación de las parejas con una base de datos de simulación.
- Se calculan probabilidades de amplificación en función de la hibridación anterior, dichos resultados se condensan por taxa y se le presentan al usuario.

Para efectos de la implementación llevada a cabo en el alcance de esta memoria, se tomaron valores por defecto de entrada y las salidas finales no han sido portadas a un formato adecuado, pues el trabajo estuvo centrado en **diseñar e implementar el núcleo del sistema.**

Capítulo 6

Extensiones

Como se abordó en los capítulos anteriores, las ideas que se quieren llevar a cabo con el sistema son muchas y de distintos alcances, algunas inmediatas, urgentes, de largo alcance, etc. Sin embargo era altamente costoso realizar cualquier modificación o extensión al sistema en su versión pasada, y por ello se sustenta esta memoria: poder brindar un núcleo de diseño e implementación que permita materializar los requerimientos y extensiones.

Por ello, en este capítulo abordaremos la multiplicidad de extensiones para el sistema, esquematizadas en tres categorías:

- Extensiones directas: Son las que se realizarán de inmediato al cierre de la iteración que define el trabajo expuesto en esta memoria, por su relevancia y urgencia para contar con un primer producto completo y cerrado.
- Extensiones deseables: De la discusión bibliográfica realizada y el *feedback* recibido de quienes han trabajado y continúan usando el sistema hasta el día de hoy, se vislumbraron extensiones que son esperables a desarrollar en el mediano plazo.
- Extensiones adicionales: Son extensiones que fueron emergiendo durante el diseño e implementación del sistema.

Se expondrán estas extensiones en orden decreciente de detalle. El objetivo de esta exposición es describir la relevancia de las extensiones, comentar algunos aspectos de su implementación e implicancias a futuro.

6.1. Extensiones directas

6.1.1. Entorno de ejecución y pulido

Tal como se expuso en secciones previas, para efectos del desarrollo se ocupó un computador portátil para el desarrollo, y todos los programas externos utilizados se instalaron en la máquina localmente para ir testeando todo dentro de la misma máquina de desarrollo. Por ello, muchos detalles como los directorios que guardan los *paths* de archivos, las ubicaciones de los programas externos, entre otras cosas, están escritas “en duro”, y por tanto se hace necesario abstraer todo esto en un **Entorno de ejecución**.

El objetivo de este Entorno de ejecución es permitir que todas las llamadas a programas externos, directorios, variables de ambiente, etc. se mantengan acotadas en un sólo lugar y el resto de los módulos acceda a ellos mediante interfaces a definir. De este modo, se brinda la posibilidad de configurar el sistema y su operación en ambientes distintos y eventualmente, de forma automática.

También, es necesario abordar una serie de mejoras y cambios al código que permitan pulir su funcionalidad:

- Incorporación de logs: Con formatos bien definidos, información suficiente y descriptiva que permitan tener una visión clara de los pasos en que se está ejecutando el sistema que permitan analizar posibles fallas y mejoras futuras.
- Re implementación de interfaces, clases, módulos: Para algunas de las interfaces, clases y módulos revisados en los capítulos anteriores, fue considerado pertinente modificarlos de manera acotada para que el diseño fuese más consistente y coherente.
- Mejoras de desempeño: Al abordar problemas de diseño relativamente pequeños, no fue necesario abordar consideraciones de eficiencia para el sistema. No obstante, dado que en un entorno de trabajo usual se tienen entradas medianas y (posiblemente muy) grandes, es necesario visitar el código en varias secciones que permitan mejorar la eficiencia de las estructuras de datos usadas, los algoritmos, las escrituras a archivo, entre otros.
- Diseñar e implementar tests: Tests para cada **Enhancer** y cada clase, además de algunos tests de esfuerzo, integración e integridad de los datos manejados, que se ejecuten durante el desarrollo y futuras iteraciones del sistema para asegurar la calidad continua del sistema.
- Estandarización de archivos: En formatos interoperables y/o visualizables en herramientas externas de uso frecuente, como **Excel**.

- Creación de filtros contenedores: El esquema general de **Enhancers** expuesto a veces deja espacio para simplificaciones mediante creaciones de **Enhancers** contenedores que engloban instancias de **Enhancers** que están ligados semánticamente de forma frecuente.
- Validación de formulación del sistema con los usuarios: Para corroborar que las fórmulas, constantes, parámetros, etc. que se usan están correctos.
- Mejora en la documentación: El sistema está largamente documentado usando **docstrings**, los cuales son *strings* que van en el código y permiten utilizar herramientas automáticas para generar documentación. Sin embargo, falta estandarizar algunas cosas, verificar que las herramientas consideran todo lo escrito y que el código está cubierto por completo con una documentación mínima.

Todos estos ítemes de pulido son abordables de forma adecuada y directa con el sistema ya implementado y permitirían contar con un núcleo del sistema que es fácil de mejorar y corregir (por los tests y *logs*), eficiente, instanciable en distintos entornos de hardware y software, válido y comunicable.

6.1.2. Paralelización

Una extensión de relevancia crítica es poder ejecutar el sistema en un entorno de ejecución paralelo. El problema es considerado altamente paralelizable debido a que el conjunto de *primers* inicial, o incluso las bases de datos a partir de las cuales generarlos, son particionables en trozos más pequeños, y muchos de los cálculos que se hacen sobre las secuencias no son dependientes del resto de los *primers*. Además, en caso de necesitarse contexto del resto de los *primers*, éste no es tan grande pues se reduciría a realizar una operación de tipo *merge* entre los *primers* que estén en distintos nodos, procurando mantener al mínimo la comunicación necesaria entre nodos. Finalmente, dado el orden de magnitud de secuencias con las cuales se espera trabajar, no sólo es pertinente paralelizar sino que también necesario.

La extensión anterior resulta clave para todo esta extensión: el entorno de trabajo tiene que ser idéntico en cada nodo o núcleo del *cluster* o máquina que se esté utilizando, o bien tiene que ser accesible de forma transparente para todos los nodos. Por ello se considera como condición necesaria contar con el entorno de ejecución para desarrollar la de paralelización.

El que el sistema se ejecute en paralelo debiese ser transparente para los diseñadores de *primers*, y debiese ser activado y desactivado en demanda por parte de los diseñadores de estrategias, siendo los cálculos que se realizan entre ambos estados transparentes para dicho usuario. Esto requiere:

- Definir un protocolo de partición y unificación de los datos, que se comunica con la interfaz definida para el Entorno de ejecución.

- Diseñar e implementar interfaces de ejecución paralela para todos los **Enhancers**.
- Testear las implicancias de trabajar en un entorno como éste, documentar y comunicar al usuario de posibles problemas en tiempo de ejecución y posiblemente incorporar cambios a los dos ítemes anteriores en función de esto.

A modo exploratorio, se estudió la librería de Python para ejecución de programas en paralelo **Parallel Python**¹, la cual permite ejecutar **funciones** de forma paralela en forma transparente para un usuario. La forma en que opera consiste en ejecutar un *script* llamado `ppserver.py` en cada nodo que se desee utilizar, y en un nodo que funciona como **host** se define el listado de los nodos en que hay una instancia de `ppserver.py` corriendo, y se invoca una ejecución en paralelo llamando a una función que recibe como argumentos una función a ejecutar en paralelo y sus argumentos, entre otros argumentos opcionales.

Se ejecutaron algunas pruebas de concepto y se consideró que el uso de esta librería podría aliviar enormemente la ejecución en paralelo del sistema. Bastaría tener los pasos descritos previamente e identificar las condiciones en que se ejecutan las funciones en paralelo para asegurar que el comportamiento dictado por la librería **Parallel Python** para las funciones es acorde a lo esperado.

6.1.3. Interfaz web

Tanto como para posibilitar el uso de la herramienta de forma acotada y amigable a los diseñadores de *primers*, como para tener un espacio en el cual mostrar la herramienta a otros investigadores en forma rápida y remota y así capitalizar la experiencia de este desarrollo para todos los involucrados, es que se considera adecuado y necesario desarrollar una interfaz web para el sistema.

La interfaz puede ser tan simple como la de **PrimerHunter**(6.1) o más enriquecida como la de **Primer3Plus**(6.2).

El nivel de detalle de la interfaz dependerá si en un comienzo es para brindar soporte a los usuarios del LBMG o exponer el trabajo realizado en una publicación científica. En cualquier caso, se debe desarrollar una aplicación web básica que extienda al sistema desarrollado en esta memoria. Para ello, se estudió el *framework* de desarrollo web **Django**², el cual permite realizar aplicaciones web usando Python. Se logró levantar en poco tiempo un prototipo para la interfaz web, por lo cual se considera que vincularlo con el sistema será una tarea relativamente directa, con un tiempo de desarrollo acotado que permita tener una interfaz simple pero pulida y que opere bien con el sistema.

¹<http://www.parallelpython.com/>

²<http://www.djangoproject.com/>

Figura 6.1: Interfaz de la herramienta PrimerHunter

Figura 6.2: Interfaz de la herramienta Primer3Plus

6.2. Extensiones deseables

A continuación, se abordan algunas de las principales extensiones que se discutieron como requerimiento en secciones anteriores, y se expone brevemente como materializarlas en función del sistema diseñado e implementado en esta memoria.

- Extender y mejorar la simulación de hibridación: Actualmente considera la energía libre de Gibbs de la última letra del *primer* pegada a una base de la secuencia, sin considerar cómo pega el resto de las bases del *primer*. Se espera poder ahondar en este cálculo y también proveer una mejor manera de condensar los resultados para un gran número de secuencias y de taxonomías distinguiendo los casos de baja probabilidad pero alta cobertura, del de baja cobertura con alta probabilidad, entre otros. Para realizarlo sólo habría que modificar los **Enhancers** asociados o crear nuevos e incorporarlos en alguna **DesignStrategy**.
- Comparación de herramientas de diseño de *primers*: Con motivo de validar el sistema ideado en el LBMG, es altamente deseable comparar el desempeño del *pipeline* con el de otras herramientas usadas actualmente en la comunidad bioinformática, eventualmente emulando unas con otras. De esta forma se brinda soporte a una publicación en caso de realizarse y permite validar la herramienta ante interesados en utilizarla. Para llevar a cabo esta comparación, se requeriría identificar las herramientas contra las cuales comparar y definir un contexto biológico sobre el cual tomar aleatoriamente inputs, pasarlos por las herramientas y compararlos según alguna medida de desempeño de las salidas, la cual puede ser éxito de amplificación o algún criterio *in silico*. Finalmente, se puede calcular un test básico para dar soporte estadístico a la comparación.
- Incorporar *primers* exitosos: Con datos obtenidos experimentalmente es posible tener *primers* exitosos con los cuales partir el diseño. Éstos servirían tanto como para validar el sistema como para eventualmente diseñar mejores *primers*. Su incorporación conlleva agregar un lugar donde incorporar dicha entrada y modificar ligeramente el algoritmo de generación de *primers* o *parejas*.
- Realizar ajuste de parámetros: Para contar con un sistema de diseño de *primers* que brinde los resultados que se esperan es deseable ajustar los parámetros en función de datos, sobre todo para los parámetros que no tienen una interpretación física directa. Esto conllevaría realizar una ejecución sistemática de múltiples instancias del sistema, sometándolo a un alto nivel de exigencia, lo cual se espera poder realizar una vez finalizado el trabajo de las extensiones directas expuestas.

6.2.1. Extensiones adicionales

En ésta sección exponemos las extensiones originadas durante el diseño e implementación del sistema, que no fueron ya previamente descritas. Se describen sin ningún orden particular entre ellas.

- Uso de `Sphinx`³ para la documentación: Esta herramienta permite generar documentación de forma automática, pero presenta la ventaja de ser la más flexible, completa y utilizada por la comunidad de desarrolladores `Python`, lo cual hace deseable su uso para la documentación del sistema.
- Diseño e implementación de librería de diseño de *primers*: En el sentido de formalizar como una librería de software tradicional todo lo desarrollado pensando en los usuarios desarrolladores del sistema en esta memoria descrito. Ello conllevaría trabajar en una serie de mejoras, simplificaciones y consideraciones adicionales que permitan usarlo como otras librerías, pero que darían mucha más exposición al sistema.
- Emulación de otras herramientas de diseño de *primers*: En una línea similar a lo anterior
- Incorporación de visualizaciones: Tanto de las métricas calculables a los objetos `PrimerSet` y `PrimerPairSet`, como de los atributos de los *primers*. Esta extensión está inspirada en las visualizaciones de la herramienta `Matlab`.

³<http://sphinx.pocoo.org/>

Capítulo 7

Trabajo Futuro y Conclusiones

7.1. Trabajo futuro

Por construcción y objetivos, este trabajo de memoria tiene mucho trabajo futuro. Sin embargo, vale la pena destacar de forma separada el trabajo futuro inmediato de valor científico. De lo visto en la revisión bibliográfica, las herramientas que actualmente se publican se encargan de innovar ya sea en un aspecto local del diseño de *primers*, o de idear una aplicación novedosa para nuevos problemas. No obstante, en el LBMG se incorporaron al menos cuatro innovaciones -uso de organismo de referencia, región SDSS, aplicación metagenómica y uso de una conceptualización inspirada en el área de recuperación de la información- en un momento en que aún no se cubrían cada una de ellas por separado. Por el valor que implicaría para el LBMG y el resultado del trabajo de esta memoria, fuertemente guiada por el objetivo de publicar, es que este trabajo futuro se debe realizar.

Además, la abstracción hecha en “filtros” permite pensar en una herramienta mucho más general de diseño de secuencias, que por construcción es flexible y extensible. La creación de una librería de diseño de *primers*, comparación y emulación con otras herramientas van en la dirección de publicar científicamente **el diseño y la arquitectura** de este sistema, pues la abstracción es en sí misma una innovación que no existe actualmente y que permitiría tener un esquema unificado para diseño de futuras aplicaciones, de alto impacto.

7.2. Conclusiones

En función de los objetivos generales propuestos:

- Tener las mismas funcionalidades de cálculo que el sistema existente.

- De uso accesible e intensivo.
- Mejoras básicas operativas.
- Diseño e implementación extensibles.

podemos decir que en cada uno de los objetivos se avanzó de forma satisfactoria. En el primero, se recogen todas las funcionalidades del sistema existente salvo un último filtro y paralelización. Como el tema de cálculo distribuido requiere de revisiones más detalladas y que fueron dejadas fuera del alcance de este trabajo, se considera que respecto al alcance del primer objetivo en particular, se progresó hasta un 95 %.

En el segundo objetivo, se constató la factibilidad de usarlo de forma intensiva, pues el *pipeline* completo se puede ejecutar desde la entrada al sistema hasta el final, cosa que no era posible con el sistema anterior. Lo que faltó realizar es una interfaz más amigable que el código del `script` de test con que se cuenta, por tanto se estima que este objetivo avanzó hasta un 75 %.

Las mejoras operativas se dedujeron de forma emergente durante la implementación del sistema, pues muchas de ellas fueron iniciativas nuevas que no eran consideradas por los usuarios actuales del sistema pues el proceso para ellos se encontraba “automatizado” en el aprendizaje generado en ellos del uso que le han dado. Este objetivo está íntimamente ligado con el anterior, siendo ambos sintetizados en “uso eficaz”, pero se separaron por ser el tercero mejoras nuevas al proceso fruto de la revisión de un usuario externo al sistema actual. En ese sentido se considera que este objetivo se logró a un 100 %.

Finalmente, el último objetivo es el que más tiempo y trabajo tomó, y sobre el cual se cimenta necesariamente todo el resto de los objetivos específicos discutidos y lo necesario para contar con una herramienta operativa. Por ello, se considera que este objetivo también se logró en un 100 %.

Así, podemos decir que para el alcance fijado en esta memoria de diseñar e implementar un **núcleo** del sistema, los objetivos se lograron a aproximadamente un 92,5 %, quedando poco trabajo por delante para contar con una herramienta operativa para el LBMG. Cabe destacar de todos modos que, para los efectos del uso futuro intensivo se requiere abordar el tema de paralelización lo cual con el conocimiento obtenido a la fecha de escritura del presente documento, no se discierne con claridad una cuantificación del porcentaje de avance en este punto. No obstante, actualmente es posible ocupar la herramienta en un entorno de línea de comandos de forma exitosa para probar distintas estrategias de diseño de *primers*, contando con un computador medianamente potente y un tamaño de datos acotado.

Se concluye que el trabajo realizado representa un avance importante en la sistematización del diseño de partidores para PCR sobre la cual basar trabajos futuros de extensión, aplicación a distintos dominios biológicos y como muestra de desarrollos concretos realizados íntegramente en el LBMG.

7.2.1. Observaciones finales

Una primera observación es la importancia de usar de forma mixta lo que se conoce como esquemas de trabajo “*top down*” y “*bottom-up*”. El primero busca entender un problema desde las capas más altas hacia abajo, el segundo intenta resolver pequeños problemas primero e ir escalando hacia arriba. El sistema fue originalmente desarrollado siguiendo un esquema del tipo “*bottom up*”, el cual era lo mejor posible considerando que se buscaba resolver un problema concreto con aplicaciones industriales que no había sido estudiado científicamente, por tanto era necesario escalar desde resultados aunque pequeños pero concretos hasta cosas más complejas, una vez que todos quienes trabajaban en el proyecto fueron aprendiendo más sobre el problema que tenían en sus manos. El aprendizaje generado y la relevancia del problema originaron muchas ideas más y una serie de problemáticas, que con el tiempo se hicieron inmanejables y motivaron este trabajo de memoria. Para compensar la experiencia que ya se tenía, se adoptó un esquema mixto de tipo “*top down*” entre el trabajo realizado en la primera etapa de identificar las componentes del sistema, el problema a abordar y la revisión bibliográfica. Así, se logró llegar a una conceptualización muy general para el diseño de *primers* que implicó la arquitectura del sistema y la forma de desarrollarlo.

Del mismo modo, se adoptó un esquema mixto para el diseño e implementación, pues ambos esquemas permiten por un lado tener claridad de lo que se debe hacer y por qué, y por otro tener resultados concretas en forma creciente, además de retroalimentarse entre sí. Esto se particulariza en el tema de diseño de software de forma especial: la importancia de implementar al diseñar.

Una segunda observación de importancia es lo fértil que es la bioinformática en problemas relevantes, complejos y de alto impacto, así como también su multidisciplinariedad. En la herramienta y problemática particular estudiada y trabajada en esta memoria se puede ver como emergen las tres grandes disciplinas de la bioinformática: El crítico conocimiento biológico para crear nuevas técnicas, identificar prioridades y sensibilidades que estudiar con más profundidad, el alto nivel de datos con que se debe trabajar que requieren de todo el soporte que brindan las ciencias de la computación y el gran número de problemas matemáticos que son formulables en todas las partes del sistema, como los cálculos termodinámicos y procesos físicos, clasificaciones estadísticas, etc.

Finalmente, el autor de esta memoria considera importante destacar lo valiosa que ha sido su formación tanto en ciencias de la computación como en matemáticas. En un trabajo eminentemente multidisciplinario como el expuesto, habiendo abordado problemas que van desde el diseño y arquitectura de software y su implementación hasta modelos termodinámicos y estadísticos, es difícil identificar si la capacidad de análisis, abstracción, formulación rigurosa de modelos, uso de herramientas tecnológicas existentes, fijar prioridades y alcances en función de múltiples objetivos, balanceo de *trade-offs* y comunicación de resultados, provienen de una u otra formación, concluyendo satisfactoriamente que el camino de formación elegido es, en efecto, uno de excelencia.

Bibliografía

- [1] K.A. Abd-Elsalam. Bioinformatic tools and guideline for pcr primer design. *African Journal of Biotechnology*, pages 91–95, 2003. [10](#)
- [2] Maria Anisimova and Olivier Gascuel. Approximate Likelihood-Ratio Test for Branches: A Fast, Accurate, and Powerful Alternative. *Syst Biol*, 55(4):539–552, 2006. [17](#)
- [3] Michael Bekaert and Emma C. Teeling. UniPrime: a workflow-based platform for improved universal primer design. *Nucl. Acids Res.*, 36(10):e56–, 2008. [17](#)
- [4] R. Boutros, N. Stokes, M. Bekaert, and E. C. Teeling. Uniprime2: a web service providing easier universal primer design. *Nucleic acids research*, 37(Web Server issue):gkp269+, July 2009. [18](#)
- [5] Bruno Contreras-Moreira, Bernardo Sachman-Ruiz, Iras Figueroa-Palacios, and Pablo Vinuesa. Primers4clades: a web server that uses phylogenetic trees to design lineage-specific pcr primers for metagenomic and diversity studies. *Nucleic Acids Research*, 37(Web-Server-Issue):95–100, 2009. [15](#), [16](#)
- [6] C.W. Dieffenbach, T. M. Lowe, and G.S. Dveksler. General concepts for pcr primer design. *PCR Methods and Applications*, Cold Spring Harbor Laboratory, 1993. [7](#)
- [7] Jorge Duitama, Dipu Mohan Kumar, Edward Hemphill, Mazhar Khan, Ion I. Mandoiu, and Craig E. Nelson. PrimerHunter: a primer design tool for PCR-based virus subtype identification. *Nucl. Acids Res.*, page gkp073, 2009. [19](#)
- [8] Robert C. Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucl. Acids Res.*, 32(5):1792–1797, March 2004. [16](#)
- [9] J. Felsenstein. Phylip (phylogeny inference package), version 3.57 c. *Seattle: University of Washington*, 1995. [16](#)
- [10] J. Fredslund, L. Schauser, L. H. Madsen, N. Sandal, and J. Stougaard. Prifi: using a multiple alignment of related sequences to find primers for amplification of homologs. *Nucleic acids research*, 33(Web Server issue), July 2005. [16](#)

- [11] Michael D. Gadberry, Simon T. Malcomber, AndrewÑ. Doust, and Elizabeth A. Kellogg. Primaclade—a flexible tool to find conserved PCR primers across multiple species. *Bioinformatics*, 21(7):1263–1264, 2005. [16](#)
- [12] Omar J. Jabado, Gustavo Palacios, Vishal Kapoor, Jeffrey Hui, Neil Renwick, Junhui Zhai, Thomas Briese, and W. Ian Lipkin. Greene SCPrimer: a rapid comprehensive tool for designing degenerate primers from multiple sequence alignments. *Nucl. Acids Res.*, 34(22):6605–6611, 2006. [16](#)
- [13] SimonÑ. Jarman. Amplicon: software for designing pcr primers on aligned dna sequences. *Bioinformatics*, 20(10):1644–1645, 2004. [17](#)
- [14] Namshin Kim and Christopher Lee. Qprimer: a quick web-based application for designing conserved pcr primers from multigenome alignments. *Bioinformatics*, 23(17):2331–2333, 2007. [16](#)
- [15] Anna-Lena Lamprecht, Tiziana Margaria, Bernhard Steffen, Alexander Sczyrba, Sven Hartmeier, and Robert Giegerich. Genefisher-p: variations of genefisher as processes in bio-jeti. *BMC Bioinformatics*, 9(S-4), 2008. [16](#)
- [16] Markus Leber, Lars Kaderali, Alexander Schönhuth, and Rainer Schrader. A fractional programming approach to efficient dna melting temperature calculation. *Bioinformatics*, 21(10):2375–2382, 2005. [20](#)
- [17] Tobias Mann, Richard Humbert, Michael Dorschner, John Stamatoyannopoulos, and William Stafford Noble. A thermodynamic approach to pcr primer design. *Nucl. Acids Res.*, 37(13):e95–, 2009. [18](#)
- [18] Nicholas R. Markham and Michael Zuker. Unafold. pages 3–31. 2008. [26](#)
- [19] Fumihito Miura, Chihiro Uematsu, Yoshiyuki Sakaki, and Takashi Ito. A novel strategy to design highly specific pcr primers based on the stability and uniqueness of 3'-end subsequences. *Bioinformatics*, 21(24):4363–4370, 2005. [13](#), [19](#), [24](#), [51](#)
- [20] C.Ñotredame, D. G. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of molecular biology*, 302(1):205–217, September 2000. [17](#)
- [21] John Rachlin, Chunming Ding, Charles Cantor, and Simon Kasif. Muplex: multi-objective multiplex pcr assay design. *Nucleic Acids Research*, 33(Web-Server-Issue):544–547, 2005. [16](#)
- [22] Timothy Rose, Jorja Henikoff, and Steven Henikoff. CODEHOP (COnsensus-DEgenerate Hybrid Oligonucleotide Primer) PCR primer design. *Nucl. Acids Res.*, 31(13):3763–3766, 2003. [16](#)

- [23] Julio Rozas, Juan C. Sanchez-Delbarrio, Xavier Messeguer, and Ricardo Rozas. Dnasp, dna polymorphism analyses by the coalescent and other methods. *Bioinformatics*, 19(18):2496–2497, December 2003. [17](#)
- [24] S. Rozen and H. Skaletsky. Primer3 on the www for general users and for biologist programmers. *Methods in molecular biology (Clifton, N.J.)*, 132:365–386, 2000. [17](#)
- [25] David Russell, Hasan Otu, and Khalid Sayood. Grammar-based distance in progressive multiple sequence alignment. *BMC Bioinformatics*, 9(1):306, 2008. [18](#)
- [26] Joseph Sambrook and David W. Russel. *Molecular Cloning: A Laboratory Manual (3rd ed.)*, Chapter 8: *In vitro Amplification of DNA by the Polymerase Chain Reaction*. Cold Spring Harbor Laboratory Press, 2001. [5](#), [6](#)
- [27] J. SantaLucia and D. Hicks. The thermodynamics of dna structural motifs. *Annu Rev Biophys Biomol Struct*, 33:415–440, 2004. [13](#), [20](#), [40](#)
- [28] Heiko A. Schmidt, Korbinian Strimmer, Martin Vingron, and Arndt von Haeseler. TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*, 18(3):502–504, 2002. [16](#)
- [29] Jason E. Stajich, David Block, Kris Boulez, Steven E. Brenner, Stephen A. Chervitz, Chris Dagdigan, Georg Fuellen, James G. Gilbert, Ian Korf, Hilmar Lapp, Heikki Lehtvaslaiho, Chad Matsalla, Chris J. Mungall, Brian I. Osborne, Matthew R. Pockock, Peter Schattner, Martin Senger, Lincoln D. Stein, Elia Stupka, Mark D. Wilkinson, and Ewan Birney. The bioperl toolkit: Perl modules for the life sciences. *Genome Res.*, 12(10):1611–1618, October 2002. [16](#), [17](#)
- [30] D. Swafford. Paup*. phylogenetic analysis using parsimony (*and other methods). version 4., 2002. [17](#)