

UNA PROPUESTA MULTIALGORÍTMICA PARA EL ANÁLISIS DE SECUENCIAS BIOLÓGICAS

Por

Wilmer Edicson Garzón Alfonso

Tesis sometida en cumplimiento parcial de los requerimientos para el grado de

MAESTRÍA EN CIENCIAS

en

INGENIERÍA DE COMPUTADORAS

UNIVERSIDAD DE PUERTO RICO
RECINTO UNIVERSITARIO DE MAYAGÜEZ

2012

Aprobada por:

Rogelio Palomera Garcia, Ph.D
Miembro, Comité Graduado

Fecha

Jaime Ramírez Vick, Ph.D
Miembro, Comité Graduado

Fecha

Fernando Vega, Ph.D
Miembro, Comité Graduado

Fecha

Jaime Seguel, Ph.D
Presidente, Comité Graduado

Fecha

Mercedes Ferrer, M.E
Representante de Estudios Graduados

Fecha

Pedro Rivera, Ph.D
Director del Departamento

Fecha

Abstract of Disertación Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Science

**A MULTI-ALGORITHM APPROACH FOR BIOSEQUENCE
ANALYSIS**

By

Wilmer Edicson Garzón Alfonso

2012

Chair: Jaime Seguel, Ph.D

Major Department: Electrical and Computer Engineering

With the completion of the Human Genome Project, between 20,000 and 25,000 genes related to human beings were sequenced, garnering more than 90 % of the DNA (deoxyribonucleic acid) human genome sequences. These are composed of letters that represent four possible nucleotides: Adenine, Cytosine, Guanine and Thymine. Also, the genetic composition of several non-human species was defined, generating an exponential growth in biological databases. As a result, it was necessary to apply computational methods in order to obtain information about the structures and biological evolution between these species.

This research presents a multi-algorithmic approach for biological sequence analysis based on statistical and computational methods. These allow for an exhaustive analysis of DNA sequences, offering different perspectives and thus allowing for a better analysis of the information.

During the course of this research, the DNA sequences of specimens that appeared to be related to the *Debaryomyces Hansenii* yeast were used. These were analyzed by the different methods that were designed through the course of this investigation. The combination of these methods allowed for a simultaneous analysis of the sequences, to then find the genetic regions of interest.

The end result of this investigation is the achievement of possible prediction of genes contained in the sequences of the two specimens mentioned above. The exhaustive search was conducted only in some regions of the sequences, due to the high demand of computational resources required to run the gene prediction method proposed here.

Resumen de Disertación Presentado a Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
Requerimientos para el grado de Maestría en Ciencias

UNA PROPUESTA MULTIALGORÍTMICA PARA EL ANÁLISIS DE SECUENCIAS BIOLÓGICAS

Por

Wilmer Edicson Garzón Alfonso

2012

Consejero: Jaime Seguel, Ph.D

Departamento: Ingeniería Eléctrica y Computadoras

Con la finalización del Proyecto Genoma Humano se logró secuenciar entre 20,000 y 25,000 genes relacionados con el ser humano, obteniendo más del 90 % de las secuencias de ADN (ácido desoxirribonucleico) del genoma humano. Estas se componen de cuatro posibles letras que representan los nucleótidos: Adenina, Citosina, Guanina y Timina. También se logró definir la composición genética de algunas especies no humanas, generando un crecimiento exponencial en las bases de datos biológicas. Debido a este crecimiento, es necesario utilizar métodos para analizar estos datos y así obtener información sobre la estructura y evolución biológica entre las especies.

Esta investigación presenta una propuesta multialgorítmica para el análisis de secuencias biológicas, basada en métodos computacionales y estadísticos. Estos permiten analizar exhaustivamente las secuencias de ADN, ofreciendo diferentes perspectivas y así permitir un mejor análisis sobre la información.

Durante el desarrollo de esta investigación, se utilizaron secuencias de ADN de especímenes posiblemente relacionados con la levadura *Debaryomyces Hansenii*,

los cuales fueron analizadas por los diferentes métodos aquí diseñados. La combinación de estos métodos permitió analizar conjuntamente las secuencias, para luego encontrar las regiones genéticas de mayor interés.

Como resultado final, se logró la predicción de los posibles genes contenidos en las secuencias de dos de los especímenes mencionados anteriormente. La búsqueda exhaustiva se realizó solo en algunas regiones de las secuencias, debido a la alta demanda de recursos computacionales requeridos al ejecutar el método de predicción de genes aquí propuesto.

Copyright © 2012

por

Wilmer Edicson Garzón Alfonso

A mi hermano Cristian y a mis queridos Padres:

José Garzón y M^a Esther Alfonso,

por su apoyo y amor sin limites;

a la paciencia y al sacrificio valeroso de

Clara Rojas y al de nuestras hijas:

Sharon Nalieth y Jael Mariana

AGRADECIMIENTOS

Quiero expresar mi mayor gratitud a Dios, por permitirme realizar mis estudios de Maestría en la Universidad de Puerto Rico.

Seguido, mi más sincero agradecimiento al Dr. Rogelio Palomera por sus consejos, orientación y el apoyo incondicional durante estos dos años de mi estadía en Mayagüez.

De igual forma al Dr. Jaime Seguel por la colaboración, asesoría y por compartir sus conocimientos conmigo durante el desarrollo de mi investigación. También al Dr. Jaime Ramírez por la orientación y sus enseñanzas, y al Dr. Fernando Vega por aceptar ser parte de mi comité graduado.

Agradecimiento muy especial a Adelaida, Aracelis, Nancy, Sandy, Luis, Raymond y demás personas del departamento por toda la colaboración y la buena disposición durante mi paso por la UPRM.

A mis amigos, en especial aquellos boricuas por tantos momentos de buen compartir, de tantas risas, de muchas alegrías y de buenas experiencias que siempre estarán en mis recuerdos.

Mil gracias por el apoyo económico ofrecido por *Texas Instruments*, pues gran parte de este trabajo fue patrocinado con fondos del proyecto *TI Analog, Digital and Mixed - Signal Electronics Program at the UPRM*.

Finalmente, a toda mi familia por todo el apoyo incondicional que siempre me brindaron, muy especialmente a mis dos hijas *Sharon Nalieth* y *Jael Mariana*, quienes han sido y serán la fuente de inspiración de cada uno de mis pasos.

Índice general

	<u>pág.</u>
ABSTRACT ENGLISH	II
RESUMEN EN ESPAÑOL	IV
AGRADECIMIENTOS	VIII
Índice de tablas	XI
Índice de figuras	XIII
 1. INTRODUCCIÓN	 1
1.1. Justificación	1
1.2. Objetivos	4
1.3. Contribución	5
1.4. Contenido de la tesis	6
 2. FUNDAMENTOS TEÓRICOS Y REVISIÓN LITERARIA	 7
2.1. Descripción general	7
2.2. La especie biológica	7
2.3. Fundamentos biológicos	9
2.3.1. La célula	9
2.3.2. Los genes	9
2.3.3. El genoma	10
2.3.4. Dogma central de la biología	10
2.3.5. Ácido desoxirribonucleico (ADN)	11
2.3.6. Predicción de genes	11
2.4. Fundamentos computacionales	13
2.4.1. Cadenas de ADN	14
2.4.2. Secuenciación de ADN	17
2.4.3. Ensamblaje de secuencias de ADN	19
2.4.4. Ensambladores de secuencias de ADN	22
2.4.5. Alineamiento de secuencias de ADN	23
2.4.6. Algoritmos para el alineamiento de secuencias de ADN	26
2.4.7. Predicción computacional de genes	32
 3. IMPLEMENTACIÓN DEL SISTEMA	 39
3.1. Datos experimentales	39

3.2.	Métodos implementados	40
3.2.1.	Alineamiento de secuencias	41
3.2.2.	Elección de los costos de penalización	44
3.2.3.	Significado biológico de los alineamientos	46
3.2.4.	Clasificación de los <i>contigs</i>	48
3.2.5.	Reconstrucción de la secuencia	51
3.2.6.	Predicción de genes	54
4.	ANÁLISIS DE RESULTADOS	59
4.1.	Ensamble de los archivos	60
4.2.	Clasificación de los <i>contigs</i>	60
4.2.1.	Elección de los costos de penalización	61
4.2.2.	Significado biológico	64
4.2.3.	Alineamiento de los <i>contigs</i>	68
4.2.4.	Reconstrucción de las secuencias	73
4.2.5.	Predicción de genes	74
5.	CONCLUSIONES Y TRABAJOS FUTUROS	84
5.1.	Conclusiones	84
5.2.	Trabajos futuros	87
	APÉNDICES	88
A.	MANUAL DEL USUARIO	89
A.1.	Prerrequisitos	89
A.1.1.	Paquetes bioinformáticos	89
A.1.2.	Librerías de Python	91
A.2.	Configuración	92
A.3.	Librerías del sistema	93
A.3.1.	Model_Gap.py	94
A.3.2.	Multicore_SW.py	95
A.3.3.	Multicore_BLAST.py	96
A.3.4.	Test_Scores.py	98
A.3.5.	Merge_Several.py	99
A.3.6.	Build_Sequence.py	101
A.3.7.	Find_Genes.py	102
A.3.8.	Eval_Genes.py	103

Índice de tablas

<u>Tabla</u>	<u>pág.</u>
2-1. Cromosomas relacionados con la especie <i>Debaryomyces Hansenii</i> . . .	8
2-2. Longitudes de las secuencias de ADN asociadas con los genes de <i>D. Hansenii</i>	8
2-3. Algunos secuenciadores de ADN de segunda generación	19
2-4. Resumen de algunos de los ensambladores de ADN	23
2-5. Comparación de resultados obtenidos entre algunos ensambladores . .	23
2-6. Tipos de búsqueda ofrecidos por BLAST	31
4-1. Información sobre los <i>contigs</i> obtenidos para las dos primeras cepas . .	60
4-2. Variabilidad en los alineamientos utilizando diferentes costos de penalizaciones	62
4-3. Cantidad de <i>contigs</i> clasificados en cada cromosoma para dos primeras cepas	68
4-4. Cantidad de <i>contigs</i> clasificados con BLAST para la primera cepa . . .	70
4-5. Cantidad de <i>contigs</i> clasificados con BLAST para la segunda cepa . .	70
4-6. <i>Contigs</i> de la primera cepa clasificados a partir del <i>e-value</i>	71
4-7. <i>Contigs</i> de la segunda cepa clasificados a partir del <i>e-value</i>	71
4-8. Clasificación final obtenida para los <i>contigs</i> de las dos cepas	72
4-9. Resultados obtenidos utilizando BLAST y Smith-Waterman sobre los <i>contigs</i> de la primera cepa	73
4-10. Longitud de las secuencias reconstruidas en cada cromosoma para la primera cepa	74
4-11. Longitud de las secuencias reconstruidas en cada cromosoma para la segunda cepa	74
4-12. Información obtenida tras ejecutar HMMgene en las secuencias los cromosomas para las dos primeras cepas	76

4-13.Cantidad de regiones ORF superpuestas, contenidas en cada cromosoma para la primera cepa	78
4-14.Cantidad de regiones ORF disjuntas contenidas en cada cromosoma de la primera cepa	78
4-15.Cantidad de regiones ORF y codificantes obtenidas con el método aquí propuesto para la primera cepa	79
4-16.Datos encontrados durante la evaluación probabilística de una región ejemplo codificante <i>seq</i>	81
4-17.Información sobre las regiones encontradas que codifican información genética en la primera cepa	82
4-18.Porcentajes y cantidad de regiones encontradas de posibles genes presentes en los cromosomas de la primera cepa	83
4-19.Información sobre las regiones encontradas que codifican información genética en la segunda cepa	83
A-1. Librerías implementadas para cada uno de los métodos propuestos . .	93

Índice de figuras

<u>Figura</u>	<u>pagina</u>
2-1. Dogma central de la biología	12
2-2. Ejemplo de una región ORF	13
2-3. Proceso de “ <i>Splicing</i> ”	14
2-4. Ejemplo de una secuencia de ADN junto con las bases complementarias	14
2-5. Ejemplo de secuencia de ADN en formato FASTA	15
2-6. Ejemplo de la secuencia en formato FASTQ	16
2-7. Ejemplo de secuencia de ADN en formato FASTQ Casava 1.8.	16
2-8. Ejemplo del proceso de ensamble de fragmentos de ADN	20
2-9. Ejemplo del grafo de superposición	21
2-10. Grafo de Bruijn para un <i>read</i> , con valor $k=3$	22
2-11. Cálculo del puntaje de alineamiento	24
2-12. Matriz Identidad (EDNASIMPLE)	25
2-13. Costo lineal Vs. Costo afín	26
2-14. Ejemplo del costo lineal y afín	26
2-15. Cálculo del valor en la posición $M(i, j)$	28
2-16. Ejemplo del grafo ORF	33
2-17. Modelo de Márkov con un estado	34
3-1. Estructura de la función <i>water</i>	42
3-2. Modelo de ejecución del algoritmo Smith-Waterman	42
3-3. Estructura de la herramienta BLAST	44
3-4. Distribución de puntajes para determinar el significado biológico . . .	48
3-5. Clasificación de los <i>contigs</i> a partir del <i>e-value</i>	50
3-6. Representación del proceso de clasificación	51

3-7. Proceso de reconstrucción realizado en cada cromosoma	52
3-8. Índices obtenidos en el alineamiento	52
3-9. Casos presentes durante la reconstrucción de la secuencia	53
3-10. Ejemplo del método consenso	54
3-11. Ejemplo de solapamiento en las regiones ORFs	55
3-12. Ejemplo de exones en una región ORF	55
3-13. Ejemplo del grafo ORF	56
3-14. Información contenida en el grafo ORF	56
4-1. Procesos en el análisis de secuencias y búsqueda de genes	59
4-2. Comportamiento del valor del error con valores iniciales $d=2$ y $e=0.5$	63
4-3. Comportamiento del valor del error con valores iniciales $d=5$ y $e=0.5$	63
4-4. Distribución de los costos de penalización, utilizando los valores iniciales $d=0.5$ y $e=0.5$	64
4-5. Distribución de los costos de penalización, utilizando los valores iniciales $d=2$ y $e=0.5$	65
4-6. Distribución de frecuencias entre el <i>contig</i> Node-919577 y el Cromosoma A	66
4-7. Distribución de frecuencias entre el <i>contig</i> Node-919577 y el Cromosoma C	66
4-8. Distribución de frecuencias entre el <i>contig</i> Node-919577 y el Cromosoma F	67
4-9. Distribución de frecuencias entre el <i>contig</i> Node-606252 y el Cromosoma D	67
4-10. Resultados de la clasificación obtenida para las dos cepas luego utilizar Smith-Waterman	69
4-11. Rendimiento del servidor durante la ejecución en paralelo del algoritmo Smith-Waterman	72
4-12. Ejemplo del archivo de salida obtenido con HMMgene	76
4-13. Alineamientos obtenidos con BLAST para una región que puede corresponder a un posible gen	81
A-1. Archivo de configuración <i>params.ini</i> utilizando dos cromosomas	92

A-2. Estructura del archivo generado con <code>Model_Gap.py</code>	95
A-3. Estructura del archivo generado con <code>Multicore_SW.py</code>	96
A-4. Estructura del archivo generado con <code>Multicore_SW.py</code> cuando existe más de un alineamiento con el mismo puntaje	96
A-5. Datos obtenidos con <code>Test_Scores.py</code>	99
A-6. Estructura del archivo generado con <code>Test_Scores.py</code>	99
A-7. Ejemplo distribución de frecuencias obtenida	100
A-8. Estructura del archivo <code>.Genes</code> generado con <code>Find_Genes.py</code>	103

Capítulo 1

INTRODUCCIÓN

Con la culminación del Proyecto Genoma Humano en 2003, fueron secuenciados todos los cromosomas relacionados con la especie humana, aproximadamente entre 20,000 y 25,000; además se definió la composición genética de algunas especies no humanas. Desde entonces se han secuenciado millones de especies, hecho que ha producido un crecimiento exponencial en las bases de datos biológicas. Las secuencias allí contenidas deben ser analizadas e interpretadas con el objeto de determinar estructura, función y evolución biológica entre las diferentes especies.

1.1. Justificación

El proceso de secuenciación de ADN (Ácido Desoxirribonucleico) produce millones de fragmentos de secuencia compuestos por cuatro letras (A, C, G o T), cada una relacionada con uno de los nucleótidos: Adenina, Citosina, Guanina y Timina.

Uno de las operaciones más importantes en el análisis de los datos biológicos es la determinación del grado de similitud entre secuencias de ADN. Las secuencias que presentan un alto grado de similitud pueden ser homólogas, es decir, corresponden a especies distintas con un ancestro en común. El grado de similitud es una forma de cuantificar el parecido existente entre dos secuencias de ADN.

Computacionalmente, existen dos enfoques para el alineamiento de secuencias de ADN: el alineamiento global y el alineamiento local. El primero pretende alinear las dos secuencias en su totalidad. El segundo, intenta alinear sub secuencias o segmentos de las secuencias seleccionando los segmentos con mayor similaridad [1]. El algoritmo de Needleman-Wunsch entrega el mejor alineamiento global entre dos secuencias [2]. En cambio, el algoritmo de Smith-Waterman encuentra el mejor alineamiento local entre dos secuencias [3]. Los dos algoritmos están basados en programación dinámica, garantizando así que el alineamiento encontrado corresponde a la solución exacta de un problema de optimización. En ambos casos, el tiempo de ejecución es cuadrático, es decir, para alinear dos secuencias de longitudes m y n el tiempo está dado por $O(mn)$ [4].

Un método heurístico que produce alineamientos locales en menor tiempo en promedio, es BLAST [5] (*Basic Local Alignment Search Tool* por sus siglas en inglés). Este método encuentra un conjunto de alineamientos con mayor puntaje entre la secuencia consulta y los millones de secuencias contenidas en una base de datos biológicas, definida por el usuario. Debido al uso de heurísticas, éste método no retorna una solución exacta al problema de optimización [1].

Todos los algoritmos de alineamiento, requieren de ciertos parámetros tales como la matriz de sustitución y el costo o penalidad por la inserción de *gaps*, estos están representados por el símbolo “-”. La selección de estos parámetros altera significativamente el alineamiento, por lo tanto estos valores no pueden ser definidos de forma arbitraria. En este trabajo los costos de insertar uno o más *gaps* fueron determinados con la ayuda de un algoritmo basado en la relación probabilística propuesta por Durbin [6]. Dicho algoritmo retorna valores de penalización apropiados, en base a una muestra aleatoria de las secuencias de ADN en estudio. Un buen alineamiento no garantiza alguna relación biológica, pues es probable que este alineamiento sea

resultado del azar. Un modo de disminuir esta probabilidad es analizar el puntaje (*score*) del alineamiento a la luz de distribuciones estadísticas. En esto se usa el hecho de que los puntajes de los alineamientos locales siguen una distribución normal.

Los métodos y algoritmos diseñados para el análisis de secuencias fueron aplicados a secuencias de ADN de especímenes aparentemente relacionadas a la levadura *Debaryomyces Hansenii*. Según el Centro Nacional para la Información Biotecnológica (*National Center for Biotechnology Information o NCBI*, por sus siglas en inglés) y el Instituto Europeo de Bioinformática (*European Bioinformatics Institute o EBI*, por sus siglas en inglés), el material genético de esta levadura se estructura en siete cromosomas diferentes. En este trabajo se analizaron secuencias de ADN relacionadas con algunas cepas de esta supuesta variante de *D. Hansenii*. Estas secuencias estaban originalmente en formato FASTQ, el cual representa los nucleótidos del genoma secuenciado junto a un puntaje de calidad para dicha secuenciación. Los archivos en FASTQ fueron ensamblados utilizando la herramienta VELVET [7], la cual retorna miles de fragmentos de secuencia llamados *contigs*, para cada una de las cepas de la especie

La estrategia diseñada para obtener secuencias a partir de los *contigs* fué la de clasificarlos en los cromosomas en que está estructurada la levadura *D. Hansenii*. Esta clasificación se realizó en base a el mayor puntaje de los alineamientos entre cada *contig* y un cromosoma. Para estos alineamientos se usaron los algoritmos, BLAST y Smith-Waterman.

En el caso de BLAST, los *contigs* de la primera cepa fueron alineados contra la base de datos biológicas “nt/nr” (*Nucleotide Collection*), en la cual están contenidas las secuencias de ADN de los siete cromosomas del espécimen. Al usar BLAST, el 12 % de los 32, 510 *contigs* de la primera cepa no retornó similitud alguna con ningún

cromosoma. Además, ninguno de los *contigs* resultó alineado con el cromosoma D.

Debido a esta falta de resolución obtenida con BLAST se decidió refinar la clasificación utilizando el algoritmo de Smith-Waterman. Este algoritmo permitió la clasificación de la totalidad de los *contigs*. Sin embargo, el posible significado biológico de estos alineamientos debió ser revaluado por métodos estadísticos.

Luego de clasificar los *contigs* y obtener una secuencia de ADN para el espécimen, es necesario encontrar aquellas regiones en la secuencia que codifican información genética. En esta investigación, la búsqueda de genes se realiza examinando todas las regiones en la secuencia de ADN. Esta búsqueda exhaustiva es una técnica poco usual, debido al tiempo y a los recursos computacionales requeridos. Usualmente, por limitaciones de tiempo la predicción de probables genes se basa en métodos estadísticos.

Entre las técnicas basadas en modelos estadísticos están: modelo oculto de Márkov (*Hidden Markov Model o HMM*, por sus siglas en inglés) [8], redes bayesianas [9] y árboles de decisión [10]. Estos métodos son rápidos, pero tienen un alto porcentaje de incertidumbre. En la referencia [11] se evalúan diferentes herramientas basadas en los métodos mencionados anteriormente, las cuales relacionan un valor probabilístico a los resultados.

El método propuesto para esta búsqueda exhaustiva es la construcción y análisis de los grafos ORF. Dichos grafos se construyen en base a reglas gramaticales, lo cual provee un enfoque no estadístico para la predicción de potenciales genes.

1.2. Objetivos

El objetivo principal es probar el poder de refinamiento y solución mediante una propuesta multialgorítmica basada en métodos computacionales y estadísticos en el análisis de secuencias biológicas. Aunque cada uno de estos métodos puede

ser utilizado independientemente, la combinación de sus características, virtudes y defectos permite resolver un gran número de incertidumbres mediante la composición y análisis de sus resultados.

A continuación se presentan los objetivos específicos de esta investigación:

- Diseñar e implantar un método estadístico computacional que facilite la elección de los costos de penalización (*open y extended gap penalty*) en base a la matriz de sustitución y a las secuencias biológicas en estudio.
- Comparar los resultados de los métodos de alineamiento, Smith-Waterman y BLAST, para resolver las incertidumbres en la clasificación de los *contigs* entre diferentes cromosomas.
- Implantar un método para estimar la probabilidad de que un alineamiento sea producto del azar. El método está basado en principios estadísticos y permite estimar el potencial de que exista algún significado biológico en los alineamientos.
- Diseñar e implantar métodos que permitan la reconstrucción de secuencias a partir de los *contigs* previamente clasificados en cada cromosomas.

Luego se realiza la identificación de los posibles genes presentes en las secuencias de ADN. Esta es una búsqueda exhaustiva y está apoyada en los grafos ORF. Además, se realiza una validación probabilística de estas regiones con base en la información actual de los genes de *D. Hansenii*.

Todos estos métodos y algoritmos están implementados en el lenguaje de programación *Python*, garantizando así la portabilidad de los métodos.

1.3. Contribución

La principal contribución de esta investigación es el diseño y uso de una propuesta multialgorítmica para analizar secuencias de ADN. Dicha propuesta contempla

el uso de métodos estadísticos y computacionales que permiten analizar y encontrar información valiosa que se hace evidente en la comparación de los resultados de este método con los existentes hasta el momento.

El otro aporte de este trabajo es el uso de estructuras gramaticales para la búsqueda de genes en las secuencias de ADN. Aunque estas estructuras se conocen desde hace ya unos años, su uso se había visto restringido por el volumen exponencial de datos que generan. En este trabajo se introduce el concepto de ventanas de largo fijo para la búsqueda de genes. La introducción de este método abre las puertas para un futuro método paralelo basado en estructuras gramaticales y grafos ORF.

1.4. Contenido de la tesis

El presente documento consta de cinco capítulos, apéndices y las referencias bibliográficas citadas en él. El primer capítulo presenta una breve descripción sobre la investigación realizada. El segundo explica los antecedentes teóricos y la revisión literaria utilizada durante el desarrollo de la investigación. Además, se presentan los fundamentos biológicos y computacionales usados en el desarrollo de la investigación. El tercero, presenta en detalle cada uno de los métodos, explicando el diseño y la forma en que fueron implementados. El cuarto capítulo presenta los resultados y el análisis de los mismos, los cuales fueron obtenidos tras la ejecución de cada uno de los métodos. Finalmente, el capítulo quinto presenta las conclusiones y los trabajos futuros relacionados con base en esta investigación. último

Capítulo 2

FUNDAMENTOS TEÓRICOS Y REVISIÓN LITERARIA

2.1. Descripción general

El principal objetivo de este capítulo es explicar brevemente algunos conceptos relacionados con el desarrollo de esta investigación. En primer lugar se describe la especie biológica que se utilizó como caso de estudio. Luego se realiza una breve explicación de algunos conceptos biológicos y fundamentos computacionales relacionados con el análisis de secuencias biológicas.

2.2. La especie biológica

Los datos biológicos analizados como caso de estudio durante la investigación, corresponden a secuencias de ADN de especímenes posiblemente relacionados con la levadura *Debaryomyces Hansenii*. Esta levadura abunda en aguas marinas y tiene la capacidad de adaptarse a bajas temperaturas (criotolerante), además tolera niveles de salinidad de hasta un 24%. También, es posible encontrarla en otros ambientes como en el proceso de maduración de los quesos y en otros alimentos como: lácteos, champiñones, alimentos congelados y algunas frutas [12]. En algunos casos *D. Hansenii* se ha encontrado en infecciones superficiales en seres humanos.

Según el Centro Nacional para la Información Biotecnológica (*National Center for Biotechnology Information o NCBI*, por sus siglas en inglés) y el Instituto Europeo de Bioinformática (*European Bioinformatics Institute o EBI*, por sus siglas en inglés), el material genético de esta levadura se estructura en siete cromosomas diferentes. La tabla 2–1 presenta la longitud (*base pair o bp*, por sus siglas en inglés) de cada uno de los siete cromosomas relacionados con la especie *D. Hansenii* [13].

Cromosoma	Longitud	Id. Genbank
A	1,249,940 bp	CR382133.2
B	1,344,482 bp	CR382134.2
C	1,587,442 bp	CR382135.2
D	1,606,296 bp	CR382136.2
E	2,007,515 bp	CR382137.2
F	2,305,761 bp	CR382138.2
G	2,051,050 bp	CR382139.2

Tabla 2–1: Cromosomas relacionados con la especie *Debaryomyces Hansenii*

De acuerdo con el NCBI, actualmente están secuenciados 6642 genes de esta especie. Cada uno de estos genes está clasificado en cada uno de los siete cromosomas de la especie. En la tabla 2–2 se encuentra la información relacionada con el promedio, la mediana y la desviación estándar relacionada con la longitud de las secuencias de estos genes en cada cromosoma. En la última fila se encuentra la cantidad total de genes presentes en cada cromosoma.

	Cromosomas (bp)						
	A	B	C	D	E	F	G
Promedio A	1,418	1,344	1,463	1,365	1,382	1,419	1,418
Mediana A	1,197	1,103	1,260	1,176	1,163	1,179	1,149
D. estándar	1,018	1,048	1,076	1,071	1,128	1,108	1,114
Cantidad	666	756	827	891	1,120	1,242	1,140

Tabla 2–2: Longitudes de las secuencias de ADN asociadas con los genes de *D. Hansenii*

Al analizar en conjunto las longitudes de las secuencias de todos los genes, se obtienen los siguientes valores: el promedio 1402, la mediana: 1176 y la desviación estándar: 1176. Estos datos fueron obtenidos sin tener presente la clasificación de los genes para cada cromosoma.

2.3. Fundamentos biológicos

Esta sección es una breve descripción de conceptos biológicos necesarios para un mejor entendimiento del desarrollo de esta investigación.

2.3.1. La célula

Es la unidad más pequeña de vida y los seres vivos están compuestos de una o más células. Las células tienen la capacidad de unirse, comunicarse y coordinarse entre ellas. Algunos organismos microscópicos, como las bacterias y los protozoos, tienen una sola célula (**unicelulares**). Las plantas, los animales y los hongos son organismos formados por numerosas células (**pluricelulares**) las cuales actúan coordinadamente [14]. El núcleo celular contiene la mayor parte del material genético. Se dice que el núcleo es el centro de control de la célula, puesto que tiene la función de mantener la integridad de los genes.

Existen dos tipos de células: las **procariotas**, no tienen núcleo celular formado, por lo tanto el material genético se encuentra disperso en el citoplasma, tal como ocurre en las bacterias. Por otra parte, las células **eucariotas** tienen el núcleo celular definido. Las plantas, hongos, levaduras y animales son especies eucariotas.

2.3.2. Los genes

En el núcleo de las células se encuentran los **cromosomas**. Cada ser vivo tiene un número específico de cromosomas. Los cromosomas están formados por largas cadenas de moléculas de ácido desoxirribonucleico (ADN o *Deoxyribonucleic acid*, *DNA*, por sus siglas en inglés). Estas cadenas contienen segmentos funcionales con

información hereditaria conocidos como genes. El gen es la unidad de almacenamiento de toda la información característica del fenotipo de las especies. Además, los genes tienen la información necesaria para la síntesis de proteínas [15]. En los seres humanos hay aproximadamente 30,000 genes y la combinación de todos los genes constituye el material hereditario que se manifiesta en el fenotipo del cuerpo humano y en la preservación sus funciones vitales.

Los seres humanos tienen un cromosoma heredado del padre y otro de la madre. Toda la información contenida allí en los cromosomas se conoce como genotipo. El fenotipo, es la expresión del genotipo. Los rasgos fenotípicos son rasgos tanto físicos como de comportamiento.

2.3.3. El genoma

El genoma es el conjunto de material genético característico en una especie. En los seres vivos el genoma se encuentra en cada una de sus células, mientras que en los virus, éste se encuentra dentro de la estructura protéica llamada cápside.

El **genoma humano** es un promedio del material genético que caracteriza a la especie humana. Este está dividido en 23 pares de cromosomas con aproximadamente de 20,000 y 25,000 genes distintos [16]. En la actualidad, se ha descifrado más del 99 % de su totalidad, gracias al *Proyecto Genoma Humano* (*Human Genome Project* o *HGP*, por sus siglas en inglés), desarrollado por el consorcio público internacional y diferentes empresas privadas. Sólo una parte del genoma codifica información; la mayor parte corresponde a regiones no codificantes o a secuencias regulatorias.

2.3.4. Dogma central de la biología

El dogma central de la biología fue enunciado por Francis Crick en 1958. Este postula que sólo el ADN puede duplicarse y, por tanto, reproducirse y transmitir la información genética a la descendencia. El comportamiento de algunos virus, llamados retrovirus como el VIH, no siguen este principio, toda vez que alteran

información del ADN o la capacidad de leerla. En general, la información genética fluye en la dirección: $\text{ADN} \rightarrow \text{ARN} \rightarrow \text{proteínas}$. La Figura 2-1 representa este principio y el proceso es como sigue:

- 1.) Etapa de replicación: La información de ADN de una molécula es copiada a otra, esto se lleva a cabo en el núcleo de la célula. Esta copia le permite al ADN transmitirse a su descendencia.
- 2.) Etapa de transcripción: El ADN presente en el núcleo de la célula es copiado a una molécula de ARN, llamada ARNm (ARN mensajero).
- 3.) Etapa de traducción: La información contenida en el ARNm es interpretada por los ribosomas, dando origen a proteínas. Los ribosomas utilizan código genético universal para determinar la secuencia codificada por el ARNm. Las secuencias de aminoácidos están delimitadas por la señal de inicio (AUG) y de terminación (UAA, UAG o UGA). Luego de encontrar la señal de inicio, el ribosoma lee grupos de tres nucleótidos para sintetizar las proteínas. Cada uno de estos grupos se conoce como **codón** y determinan un aminoácido en particular [17].

La principal diferencia entre el ADN y el ARN, es que el ADN almacena información genética mientras que el ARN transporta información para la producción de proteínas.

2.3.5. Ácido desoxirribonucleico (ADN)

El ADN está compuesto por nucleótidos, cada uno está formado por un azúcar (desoxirribosa) y una base nitrogenada: adenina (A), timina (T), citosina (C) o guanina (G) y un grupo fosfato que actúa como enganche entre nucleótidos. Las bases adenina y timina, guanina y citosina son complementarias [18].

2.3.6. Predicción de genes

La identificación de los segmentos que potencialmente contienen información genética en el genoma es uno de los más grandes desafíos computacionales en la

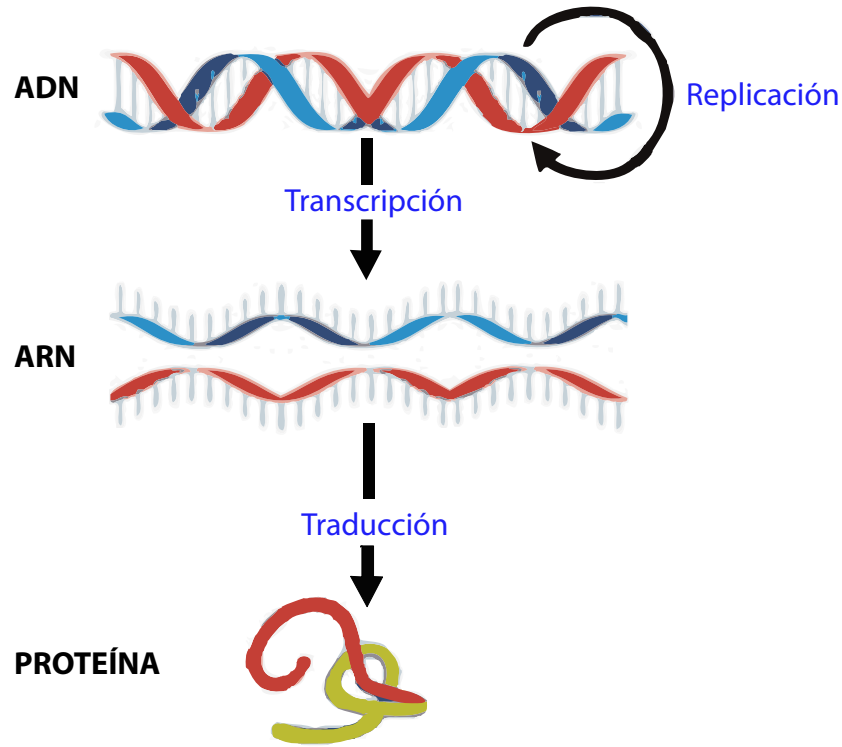


Figura 2-1: Dogma central de la biología

Bioinformática. El genoma es representando por largas cadenas de letras (A, C, G o T), las cuales corresponden a cada uno de los aminoácidos ya mencionados. En cada una de estas cadenas es necesario encontrar las regiones que codifican la información genética. Estas regiones son conocidas como “marco abierto de lectura” (*Open reading frame* o *ORF*, por sus siglas en inglés). En el caso de células eucariotas, las regiones ORF contienen exones e intrones. Los exones son las regiones de interés, pues en ellas se codifica la información genética, mientras que los intrones no contienen información relevante. La búsqueda de genes en especies eucariotas se hace significativamente más compleja, debido a la necesidad de distinguir exones de intrones. Las regiones ORF están generalmente delimitadas por un codón inicial (ATG) y un codón final (TAA, TAG o TGA) (Figura 2-2, extraída del sitio www.genome.gov).

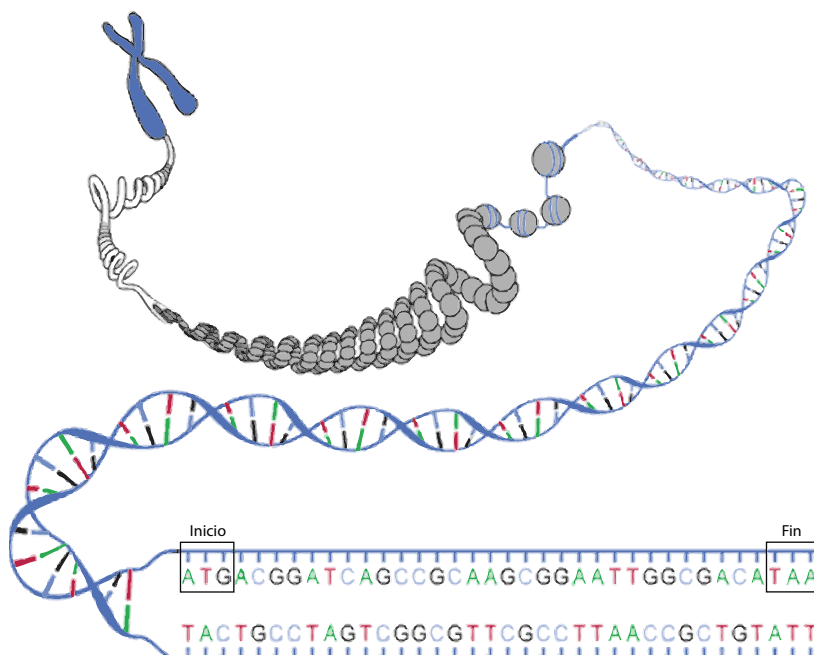


Figura 2-2: Ejemplo de una región ORF

Las secuencias de codificación (*coding sequences* o *CDS*, por sus siglas en inglés) son fragmentos del ADN compuestos únicamente de exones. Es solo esta la información que contribuye a la producción de proteínas. En las especies procariotas, las regiones ORF y CDS son las mismas, debido a la inexistencia de intrones. En las regiones ORF es posible encontrar exones: entre un codón inicial y un sitio donante (GT), entre un sitio aceptador (AG) y un sitio donador (GT), y finalmente entre un sitio aceptador (AG) y un codón final. La eliminación de los intrones produce el ARN mensajero (mARN), el cual es una versión depurada del ARN inicial. El ARNm es la concatenación de los exones que resultan de este proceso de purificación, también llamado *splicing*. La Figura 2-3 muestra este proceso, el cual ocurre en el ARNm y tiene lugar antes de abandonar el núcleo de la célula.

2.4. Fundamentos computacionales

En esta sección se presenta una breve descripción de los conceptos computacionales y algorítmicos relacionados con el desarrollo de esta investigación; estos

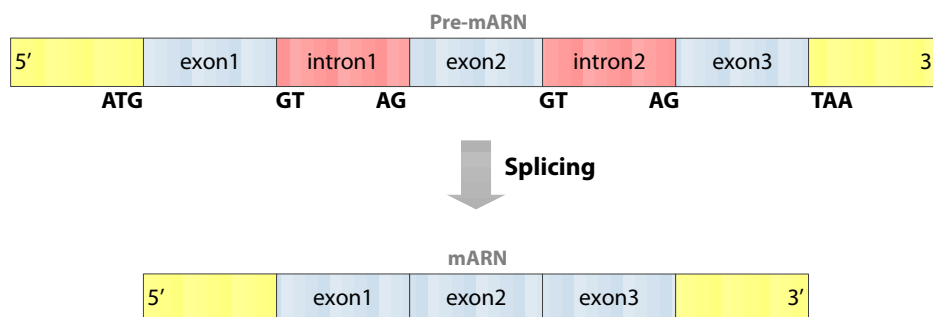


Figura 2-3: Proceso de “*Splicing*”

fundamentos permiten tener un mejor entendimiento de este trabajo en el ámbito computacional.

2.4.1. Cadenas de ADN

También conocidas como secuencias de ADN, son un conjunto de letras que representan la estructura primaria de una molécula de ADN, donde cada una de las letras representan los nucleótidos: **A**denina y **T**iamina, **G**uanina y **C**itosina. En la molécula de ADN, estos nucleótidos forman pares complementarios, dando origen a las bases nitrogenadas AT y CG. En el caso típico, las secuencias de ADN se encuentran pegadas unas a otras, sin espacios, yendo de izquierda a derecha (5' → 3'). No todas las secuencias de ADN representan información genética.

Sec. Principal :	A	T	G	A	T	T	G	A	C	C	G	A	G	G	A
Complementaria :	T	A	C	T	A	A	C	T	G	G	C	T	C	C	T

Figura 2-4: Ejemplo de una secuencia de ADN junto con las bases complementarias

Las secuencias de ADN son fundamentales en la Bioinformática; estas constituyen una abstracción matemática básica y útil de un fenómeno bioquímico. Las secuencias de ADN son obtenidas a partir de muestras del material biológico siguiendo diferentes técnicas de secuenciación. El proceso de secuenciación produce millones de segmentos de ADN, los cuales son guardados en archivos. Estos archivos están compuestos por secuencias de ADN representadas en diferentes formatos. Entre los

más usados están: EMBL, FASTA, GCG, GenBank, IG, entre otros.

Formato FASTA

Es el formato de archivo mas común para representar secuencias de ADN. La representación de la secuencia comienza con una línea de descripción y luego las diferentes líneas relacionadas a los nucleótidos que representan la secuencia.

La línea de descripción comienza con el símbolo ‘>’ en la primera columna. La palabra después de este símbolo es el identificador de secuencia (ID), y el resto de la línea es la descripción de la misma. Un ejemplo simple de una secuencia en formato FASTA se muestra en la figura 2–5. Esta secuencia es un fragmento del cromosoma A de la levadura *Debaryomyces hansenii*.

```
>gi|199431295|emb|CR382136.2| Debaryomyces hansenii CBS767 chromosome A complete sequence
AAATCGGCTTTTCCCCCTCCCTTTTTTTCCGCAATTTGGGGGGGGGGGGGGGGGGGAAAAAGGGC
AAAAAAAAACCCCCCGGGGAATTTAAACCCCTCTTGCCCATTTACAGGTGTGGGCAACTGGTTGGGG
AAGGGGGGAATGGGGGGGGGGCTTTTCGTATTATACCCCAACCGGGGAAAAAGGGGGATTGCTTCAAG
GGGGATTAAGTTGGGAACGCCAGGGGTTTTCCAGTCCAGGCGTTTGAAAAACGACGCCAAGGC
```

Figura 2–5: Ejemplo de secuencia de ADN en formato FASTA

También se encuentra el formato **multi-FASTA**, el cual permite representar diferentes secuencias de ADN en formato FASTA en un solo archivo.

Formato FASTQ

El formato FASTQ es una variante del formato FASTA, tiene la capacidad de almacenar las secuencias de ADN y un puntaje de calidad relacionado con la misma. El valor del puntaje es asignado utilizando PHRED; este es definido en términos de la probabilidad del error asociada a cada nucleótido [19]:

$$Q_{PHRED} = -10 \log_{10}(P_e) \quad (2.1)$$

La figura 2–6 muestra un ejemplo de una secuencia de ADN en formato FASTQ. La línea 1 comienza con el símbolo ‘@’ y es seguido por el identificador de secuencia

y una descripción opcional. La línea 2 es la secuencia de las letras primas. La línea 3 comienza con el símbolo ‘+’ y finalmente la línea 4 codifica los valores de calidad para la secuencia de la línea 2, y debe contener el mismo número de símbolos que las letras en la secuencia [20].

```
@identificador de la secuencia
AAATGGCAAAGCAGTATCGATCAAATAATCTTTGTTGTTACTCAACCGG
+
!"%$$&&/%++)(%(((.)))).%$#55CCF>>>>>>CCCCCCC65
```

Figura 2–6: Ejemplo de la secuencia en formato FASTQ

Formato Solexa/Illumina

Este formato es una variante del formato FASTQ, cambia la forma en que se calcula la calidad del puntaje para cada nucleótido. La compañía Solexa, líder en el proceso de secuenciación y genera los archivos calculando el puntaje de forma diferente a PHRED. El puntaje de calidad está definido por [19]:

$$Q_{Solexa} = -10 \log_{10} \left(\frac{P_e}{1 - P_e} \right) \quad (2.2)$$

Solexa realiza el proceso de secuenciación haciendo millones de reacciones en paralelo. Estas técnicas son conocidas como HiSeq y serán explicadas en la próxima sección. Durante el proceso de secuenciación, Solexa genera dos archivos en formato FASTQ, los cuales están bajo la especificación CASAVA 1.8. Estos dos archivos deben ser ensamblados y así obtener una aproximación de la secuencia original. La figura 2–7 presenta una muestra de uno de los archivos en formato Solexa/Illumina bajo la especificación mencionada anteriormente.

```
@HWI-ST156:422:D09KTACXX:5:1101:1125:1217 1:N:0:CAGATC
ATTCGGTTTGTTCACGCGTAATTCTGTTATTTGATGCAAAAAATGAAAAAGANGC
+
??@DDDBDHDHFFEC<FEHII>=F@BGBGG9DDGHBDHIIABG9:AF3=@=C9';#,,,?56;>
@HWI-ST156:422:D09KTACXX:5:1101:1307:1103 1:N:0:CAGATC
ATTTGGTAGTGAGTAATATTAGATATCCTTATAAAGAGTAGCCTGAATTAAGATCAG
+
```

Figura 2–7: Ejemplo de secuencia de ADN en formato FASTQ Casava 1.8.

2.4.2. Secuenciación de ADN

El proceso de secuenciación se compone de diferentes técnicas bioquímicas que permiten determinar el orden de las bases (A, C, G y T) presentes en la muestra biológica de ADN de la especie. Estas técnicas han evolucionado rápidamente, un ejemplo notable es el *Proyecto Genoma Humano*, el cual fue culminado en menos tiempo del planeado. Existen diferentes técnicas para secuenciar ADN; algunas de ellas se analizan a continuación.

Secuenciadores de primera generación

En los años 70, Walter Gilbert y Allan Maxam [21], en Harvard, y Frederick Sanger [22], en Inglaterra, trabajaban de forma independiente en desarrollar nuevos métodos para secuenciar el ADN. En 1975, Sanger y sus colaboradores desarrollaron el método enzimático de terminación de cadena para secuenciar el ADN, también conocido como el método didesoxi (didesoxinucleótidos). Dos años más tarde, Sanger publicó la primera secuencia de ADN, correspondiente al genoma completo del bacteriófago *PhiX174* (5368bp). Por otra parte, Gilbert y Maxam publicaron el método de fragmentación química para secuenciar el ADN. Esta técnica no tuvo éxito, era un proceso muy manual, costoso y en cierta forma peligroso, debido al uso de compuestos radioactivos. Además permitía secuenciar sólo pocos cientos de pares de bases a la semana. Por estas razones, el método de Sanger resultó ser el más popular de la época [20].

A partir de esos aportes, comenzó una verdadera competencia por la secuenciación de cientos de especies, y muchas personas comenzaron a pensar en la posibilidad de secuenciar el genoma humano. A finales de los 90, avances en automatización permitieron mejorar y aumentar el rendimiento del proceso de secuenciación. Estos secuenciadores automáticos permitían la secuenciación en pocas horas de hasta 96 muestras de ADN al tiempo, con longitudes de secuencia superiores a 500bp.

Por la rápida y continua evolución de los secuenciadores, en 1990 el mundo científico comenzó la carrera por secuenciar el ADN del ser humano, por lo tanto se firma el *Proyecto Genoma Humano* (*Human Genome Project o HGP*, por sus siglas en inglés). Alrededor de una década después, en 2001, se publicó el primer borrador del genoma humano. Este proyecto costó al rededor de 3,000 millones de dólares logrando leer aproximadamente 3,000 millones de nucleótidos, es decir (\$1/nt). Como dato significativo, al final de esta primera generación se pasó de miles de secuencias reconocidas en 1995 a 20 millones de secuencias reconocidas en el año 2002, logrando un crecimiento exponencial en las bases de datos biológicas.

Secuenciadores de segunda generación

Debido al continuo desarrollo tecnológico, muchas compañías comenzaron a investigar y a desarrollar métodos de secuenciación más rápidos y económicos. En la búsqueda para reducir costos, y aumentar la producción de resultados, surgieron los secuenciadores de ADN de alto rendimiento (*High Throughput Sequencing o HiSeq*, por sus siglas en inglés) [20]. Estos secuenciadores tienen la capacidad de generar millones de reacciones en paralelo. De esta forma, la cantidad de reactivos necesarios se minimiza reduciendo así el costo por base leída.

En la actualidad, existen cinco tecnologías de alto rendimiento líderes en el mercado. Cada una de estas trabaja de forma diferente tal como se resume en la tabla 2–3 [23, 24]. Estas tecnologías permiten obtener volúmenes de lectura en el orden de las Gigabases de forma más rápida con respecto a los de primera generación [25].

Solexa y SOLiD son las compañías que han desarrollado las técnicas más económicas, la mayor desventaja es la falta de capacidad para generar lecturas superiores a las 70–100 bases, limitando así la capacidad de secuenciar genomas de gran tamaño. Una importante ventaja de los secuenciadores de segunda generación, está en la no

Compañía	Secuenciación	Costo x Gb	Costo del instrumento	Longitud de lectura
Roche/454	Polimerasa(pirosecuenciación)	0.45	\$500,000	330bp
Illumina/Solexa	Polimerasa(reversible)	18	\$540,000	70 – 100bp
SOLiD	Ligasa	30	\$595,000	50bp
Polonator	Ligasa	12	\$170,000	26bp
HeliScope	Polimerasa(asincrónico)	37	\$999,000	32bp

Tabla 2–3: Algunos secuenciadores de ADN de segunda generación

clonación del ADN, evitando el trabajo tedioso de crear genotecas. En la actualidad, estos secuenciadores son de gran éxito, permitiendo reducir de \$10 en 1990 a \$0.01 en 2005, el costo por lectura de un nucleótido.

Secuenciadores de tercera generación

La búsqueda de reducción de costos de secuenciación y de aumentar la calidad en el proceso, ha llevado a los investigadores a buscar nuevas tecnologías. Los secuenciadores de tercera generación, se especializan en la secuenciación de una única molécula de ADN (*single molecule real time sequencing*) [26]. El primer secuenciador de esta generación fue desarrollado por *Helicos BioSciences* [27]; se basa en la secuenciación en tiempo real de miles de millones de pequeñas moléculas únicas de ADN adheridas a una superficie sólida. Debido al tamaño reducido de las lecturas generadas (entre 25 y 45 bases), esta tecnología es recomendada para la resecuenciación de genomas conocidos y no para la secuenciación de nuevas especies. Por otra parte, la compañía *Pacific Biosciences* trabaja en una técnica que promete leer hasta 1000 nucleótidos, resolviendo las limitantes de lectura de la anterior generación [24].

2.4.3. Ensamblaje de secuencias de ADN

Debido al millón de reacciones que ocurren en paralelo en los secuenciadores de segunda generación (HiSeq), se obtienen millones de fragmentos cortos (*reads*)

de secuencias de ADN. Por lo tanto, es necesario ensamblar dichos fragmentos y así tratar de obtener una aproximación a la secuencia original. Para llevar a cabo el montaje de los *reads* se utilizan los ensambladores de secuencias de ADN. Básicamente, el ensamblaje se realiza alineando y combinando conjuntamente cada uno de los diferentes fragmentos. La figura 2-8 presenta un ejemplo simple de ensamblaje, donde se realiza el alineamiento y ordenamiento de los fragmentos de la izquierda intentando construir la secuencia original.

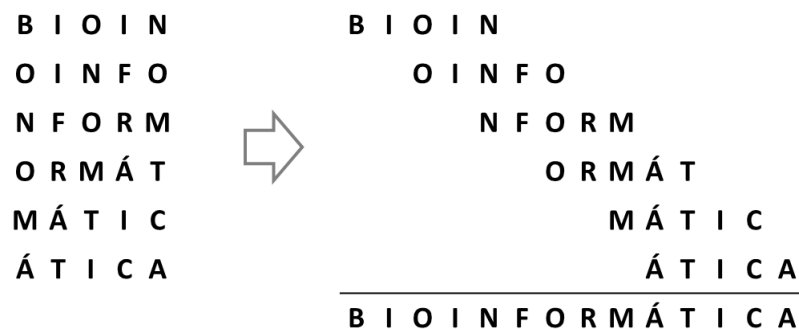


Figura 2-8: Ejemplo del proceso de ensamble de fragmentos de ADN

En la mayoría de los casos, luego del ensamblaje no es posible obtener una sola secuencia. Esto se debe a la falta de información al momento de unir los fragmentos o por algún tipo de ruido introducido al momento de la secuenciación del ADN. Por esta razón, se obtienen conjuntos de letras compuestos por diferentes fragmentos ensamblados llamados (*contigs*) y se encuentran separados por espacios en blanco. En ese caso no es posible unir los *contigs*, debido a la falta de información entre ellos.

El ensamblaje de secuencias se puede clasificar en dos tipos:

- **Mapping:** El ensamble se realiza con base a una secuencia original, tratando de conseguir la mejor aproximación al unir los fragmentos de ADN.
- **Novo:** El ensamble se realiza entre los fragmentos cortos, obteniendo secuencias de mayor longitud.

Computacionalmente, encontrar la cadena de menor tamaño que contiene a otras cadenas es un problema NP-completo. Algunos de los enfoques utilizados para resolver el problema están basados en la teoría de grafos. Para ensamblar fragmentos cortos de ADN en especies biológicas, algorítmicamente existen dos enfoques [28]:

■ Grafos de superposición (*Overlap Graphs*)

A partir de los fragmentos cortos (*reads*) se calcula la superposición entre todos y esta información se representa en un grafo, en el cual, cada uno de los nodos representan un *read* y los arcos están determinados por el solapamiento entre los dos *reads*. La figura 2-9 ilustra el grafo, para el solapamiento entre seis *reads*; la parte en color representa el solapamiento entre cada par de *reads*.

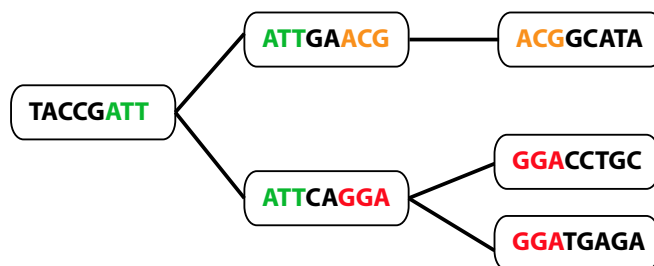


Figura 2-9: Ejemplo del grafo de superposición

■ Grafos de Bruijn

Es el enfoque más utilizado por los ensambladores de secuencias de ADN. Permite trabajar con millones de *reads* reduciendo notablemente el tiempo que tardan la construcción de los grafos de superposición. Se comienza por partir cada uno de los *reads* en cadenas de longitud (*k-mer*) para luego construir los diferentes nodos. Un arco existe entre dos nodos *a* y *b*, si existe un *k-mer* en el cual *a* es el prefijo y *b* es el sufijo. La Figura 2-10 representa el grafo de Bruijn para un *read* utilizando palabras de longitud 3, el color rojo indica los fragmentos comunes a lo largo de la secuencia. El parámetro *k* permite reducir el nivel de redundancia entre cada una de las superposiciones y tiene influencia significativa en la calidad del ensamblado.

Además, cuando se tienen millones de fragmentos, el grafo no se expande, logrando así una reducción al momento de construir y recorrer el grafo.

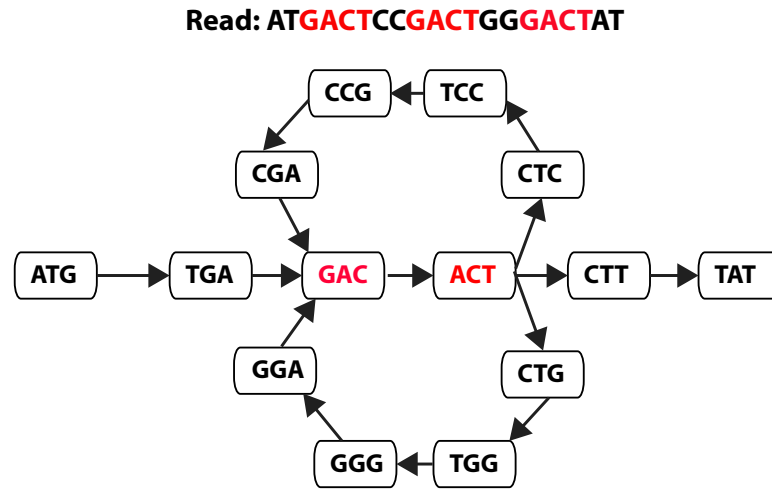


Figura 2–10: Grafo de Bruijn para un *read*, con valor $k=3$

2.4.4. Ensambladores de secuencias de ADN

Existen diversos programas para ensamblar fragmentos cortos de ADN (*reads*), los cuales están desarrollados bajo los dos enfoques explicados anteriormente. La tabla 2–4 presenta una breve descripción de algunos de los ensambladores más utilizados.

Ensamblador de secuencias de ADN: VELVET

VELVET es una de las herramientas más utilizadas en la actualidad. Permite ensamblar secuencias de ADN obtenidas por diferentes tecnologías como: Solexa, SOLiD o 454 [28]. VELVET usa los grafos de Bruijn y diferentes métodos para corregir y reducir los errores luego de la construcción del grafo. VELVET fue desarrollado por Daniel Zerbino y Ewan Birney en asocio con el Instituto Europeo de Bioinformática (*European Bioinformatics Institute o EBI*, por sus siglas en inglés) [7].

La compañía Illumina, líder en el desarrollo de tecnologías de secuenciación, comparó los cuatro ensambladores: VELVET, ABySS, Forge y SOAP-denovo [28].

Nombre	Descripción	Tipo	Fortalezas	Técnicas de secuenciación
Velvet (2009)	Basado en Bruijn, Corrección de errores luego de construir el grafo, trabaja de forma paralela	Genomas cortos	Rápido, (~30mins), fácil de utilizar, largos <i>contigs</i>	Sanger, 454, Solexa, SOLiD
Ray (2010)	Basado en Bruijn, trabaja de forma paralela	Genomas completos	Mezcla de HiSEQ, largos <i>contigs</i>	Solexa, 454
ABYSS (2011)	Basado en grafos, de Bruijn	Genomas largos	Facil de utilizar, genera largos <i>contigs</i>	Solexa, SOLiD
Forge (2010)	Basado en grafos, de superposición	Genomas	largos <i>contigs</i>	Sanger, 454, Solexa,

Tabla 2–4: Resumen de algunos de los ensambladores de ADN

En la tabla 2–5 se resumen los resultados obtenidos utilizando diferentes tamaños de *contigs*, relacionados con la bacteria *Escherichia coli*. En las diferentes comparaciones realizadas, VELVET arrojó un cubrimiento en el genoma superior al 99.72 %, siendo la herramienta que con mejores resultados.

Paquete Software	Medición N50 (bp)	Longitud Contig (bp)	Cubrimiento del genoma
Velvet 0.7.31, k=31	61,802	115,666	99,72 %
ABYSS 1.0.8, k=42	45,171	140,706	99,64 %
Forge 1.0, k=15	70,447	444,471	99,4 %
SOAPdenovo 1.0	3,026	20,258	99,51 %

Tabla 2–5: Comparación de resultados obtenidos entre algunos ensambladores

2.4.5. Alineamiento de secuencias de ADN

La mayor utilidad de las bases de datos biológicas esta en la determinación del grado de similitud entre las secuencias de ADN. Dicho nivel de similitud es interpretado como indicativo de la existencia de algún tipo de relación biológica. El alineamiento consiste en confrontar una secuencia con otra de tal forma que

sobresalgan las diferencias y similitudes entre cada uno de los pares de nucleótidos. Si las secuencias tienen alto grado de similitud hay una alta probabilidad de que sean homólogas, es decir, tienen un ancestro en común [29].

El nivel de similitud del alineamiento es cuantificado por el así llamado puntaje del alineamiento (*score*). Para tener un mejor puntaje es posible insertar *gaps* (“-”) en el alineamiento, y así tener una mejor aproximación entre las dos secuencias. La inserción de un *gap* tiene asociado un costo de penalización. Por otra parte, es necesario determinar el costo de reemplazar un nucleótido por otro (*match/mismatch*). Los dos costos anteriores se ven reflejados en el valor del *score* del alineamiento.

La figura 2–11 representa el *score* asociado a dos secuencias S_1 y S_2 , donde (α) determina el valor del *match/mismatch* para cada nucleótido y (δ) es el costo de penalización por la inserción de *gaps* en el alineamiento. El valor de reemplazar un nucleótido por otro está determinado por las matrices de sustitución que se discuten en la siguiente sección.

Sec1:	A	-	C	A	T	T	G	G	C
Sec2:	A	A	C	T	A	T	-	G	A
Score:	$\alpha_{AA} + \delta + \alpha_{CC} + \alpha_{AT} + \alpha_{TA} + \alpha_{TT} + \delta + \alpha_{GG} + \alpha_{CA}$								

Figura 2–11: Cálculo del puntaje de alineamiento

Matrices de sustitución

Determinan el valor de *match* y *mismatch* entre cada par de nucleótidos y están basadas en probabilidades obtenidas a partir de observaciones biológicas. Las entradas de la diagonal determinan el valor de coincidencia *match* entre cada par de nucleótidos y las demás entradas corresponden al valor de los *mismatches*. Existen diferentes matrices, BLOSUM (BLOck SUBstitution Matrix) y PAM (Point Accepted Mutation) que son las más utilizadas para el alineamiento de secuencias de proteínas.

Por otra parte, se encuentran las matrices EDNAMAT, EDNAFULL y EDNASIMPLE (figura 2–12) para el alineamiento de secuencias de ADN, siendo esta última la matriz más utilizada.

	A	T	C	G
A	1	-1	-1	-1
T	-1	1	-1	-1
C	-1	-1	1	-1
G	-1	-1	-1	1

Figura 2–12: Matriz Identidad (EDNASIMPLE)

Costos de penalización

Como se mencionó anteriormente, en el proceso de alineamiento se insertan *gaps* para conseguir una mejor aproximación entre dos secuencias. Este costo tiene dos formulaciones: estándar y afín.

El costo estándar está definido por la ecuación: $\gamma(g) = \lambda g$, donde λ es un valor constante y g es la longitud del *gap*. Con este tipo de costo el valor de penalización es igual para todos los *gaps*. Por esta razón es también llamado costo lineal. Por otra parte, el costo afín está determinado por la ecuación: $\gamma(g) = -d - (g - 1)e$, donde d y e son los valores de penalización del *open* y *extended gap* respectivamente. El valor del *open gap* determina el costo cuando se inserta el primer *gap* en una región del alineamiento, mientras que el *extended gap* define el costo adicional de insertar *gaps* consecutivos 2–13.

La ventaja del costo afín es una reducción en el valor de la penalización en la inserción de *gaps* consecutivos, lo cual representa más adecuadamente la división en dos mitades de una secuencia [6]. La figura 2–14 es un ejemplo simple donde se calcula el puntaje del alineamiento utilizando la matriz EDNASIMPLE y asumiendo que los valores de d y e son -3 y -1 respectivamente. Con respecto al costo de la

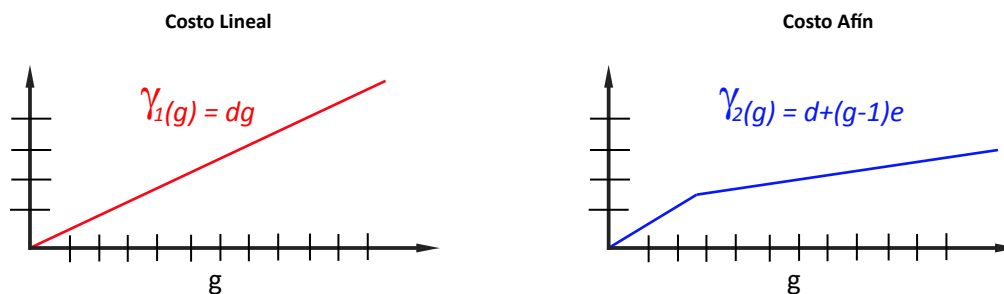


Figura 2-13: Costo lineal Vs. Costo afín

penalización se observa una reducción cuando se utiliza el afín, siendo esto una gran ventaja sobre el costo lineal.

Costo Lineal	Costo Afín
Sec1: A C G T T A C C	Sec1: A C G T T A C C
Sec2: A C G - - A - C	Sec2: A C G - - A - C
1 + 1 + 1 -3 -3 +1 -3 +1 = -4	1 + 1 + 1 -3 -1 +1 -3 +1 = -2

Figura 2-14: Ejemplo del costo lineal y afín

La elección de los valores de d y e tiene alto impacto en el puntaje del alineamiento. Por esta razón, estos valores deben ser elegidos mediante algún razonamiento específico o algún modelo matemático y/o estadístico que sustente la elección [30].

2.4.6. Algoritmos para el alineamiento de secuencias de ADN

Existen dos métodos para el alineamiento de secuencias de ADN: el alineamiento global y el alineamiento local. El alineamiento global intenta alinear las dos secuencias en su totalidad. Mientras que el alineamiento local intenta alinear sub secuencias o segmentos de menor tamaño a la secuencia. Ambos algoritmos están basados en programación dinámica, garantizando así que el alineamiento encontrado corresponde a la solución exacta de un problema de optimización.

Algoritmo de alineamiento global

El **algoritmo de Needleman-Wunsch** retorna el mejor alineamiento global entre dos secuencias. Este fue el primer algoritmo que permitió encontrar el nivel de similitud entre dos secuencias. Fue publicado en 1970 por Saul Needleman y Christian Wunsch [2], el cual permite encontrar el alineamiento con el puntaje máximo.

Para encontrar el mejor alineamiento entre dos secuencias X y Y , se construye la matriz $M_{m \times n}$, donde m y n son las longitudes de cada una de las secuencias. En la matriz M las columnas y las filas están determinadas por los nucleótidos de X y Y respectivamente. A medida que el algoritmo avanza, el valor de $M(i, j)$ será la puntuación óptima al alinear los primeros j elementos de X y los primeros i elementos de Y . Este algoritmo está representado en el pseudocódigo 1. Luego de construir la matriz se realiza el proceso de *backtracking*, el cual, a partir de la última coincidencia del alineamiento $M(i + 1, j + 1)$ comienza en retroceso la búsqueda del camino que maximice la función.

Pseudocódigo 1 (Alineamiento global entre secuencias)

Entrada: Las secuencias X y Y , el valor del *gap* λ y la matriz de sustitución D

Salida: El alineamiento global y el puntaje óptimo

Inicialización:

$M(i, 0) \leftarrow -i \cdot \lambda$, para todo $0 \leq i \leq n$

$M(0, j) \leftarrow -j \cdot \lambda$, para todo $0 \leq j \leq m$

para $i = 1$ **to** n **hacer**

para $j = 1$ **to** m **hacer**

$$M(i, j) \leftarrow \max \begin{cases} M(i-1, j-1) + D(X_i, Y_j) & \text{<coincidencia>} \\ M(i-1, j) - \lambda & \text{<eliminación>} \\ M(i, j-1) - \lambda & \text{<inserción>} \end{cases}$$

fin para

fin para

Realizar el proceso de *backtracking*

devolver *alineamiento*, *score*

En la figura 2-15 se ilustra la forma de calcular el valor en la posición $M(i, j)$. Se asume que el valor de λ es 1. Allí se se observa la inicialización de la matriz y también la dependencia de los valores adyacentes para determinar el valor de la posición actual.

		Y								
		A	C	T	C	G	A	G	...	C
	0	-1	-2	-3	-4	-5	-6	-7	...	-i
C	-1									
T	-2									
A	-3									
:	:									
G	-j									(i,j)

Figura 2-15: Cálculo del valor en la posición $M(i, j)$

Algoritmo de alineamiento local

Por otra parte, el **algoritmo de Smith-Waterman** encuentra el mejor alineamiento local entre dos secuencias [3]. En este se alinean únicamente las zonas más similares, obteniendo los segmentos con mayor similitud dentro de un contexto mayor [1]. Este algoritmo fue propuesto por Temple Smith y Michael Waterman en 1981 y en principio se basó en el algoritmo de alineamiento global. La matriz $M_{m \times n}$ es inicializada con valor cero; además el inicio y fin de un camino óptimo no necesariamente se origina desde la última fila o columna, este puede encontrarse en cualquier sitio de la matriz. Este busca en toda la matriz todas las regiones con alta similitud local. Este algoritmo está representado en el pseudocódigo 2.

El mejor alineamiento se obtiene buscando en la matriz M el valor más alto. Este puede estar en cualquier lugar de la matriz, debido a que los alineamientos locales pueden comenzar y terminar en cualquier punto. Luego se realiza el *traceback* para obtener todos los fragmentos que forman parte del alineamiento óptimo. El proceso

Pseudocódigo 2 (Alineamiento local entre secuencias)

Entrada: Las secuencias X y Y , el valor del *gap* λ y la matriz de sustitución D

Salida: El alineamiento local y el puntaje óptimo

Inicialización:

$M(i, 0) \leftarrow 0$, para todo $0 \leq i \leq n$

$M(0, j) \leftarrow 0$, para todo $0 \leq j \leq m$

para $i = 1$ to n **hacer**

para $j = 1$ to m **hacer**

$$M(i, j) \leftarrow \max \begin{cases} 0 \\ M(i-1, j-1) + D(X_i, Y_j) \\ M(i-1, j) - \lambda \\ M(i, j-1) - \lambda \end{cases}$$

fin para

fin para

Realizar el proceso de *traceback*

devolver *alineamiento*, *score*

de *traceback* finaliza cuando se encuentra un cero en la diagonal de M ; este es el comienzo del alineamiento.

En términos de aplicación biológica, este algoritmo permite identificar posibles regiones localmente conservadas, es decir, aquellas regiones del ADN donde el material genético se ha conservado a través del tiempo. El algoritmo de Smith-Waterman alinea pares de secuencias, la secuencia en cuestión (*subject sequence*) y la secuencia de consulta (*query sequence*).

Este algoritmo usa programación dinámica, garantizando así que el alineamiento encontrado corresponde a la solución exacta de un problema de optimización. El tiempo de ejecución es cuadrático, es decir, para alinear dos secuencias de longitudes m y n el tiempo es $O(mn)$ [4].

Los algoritmos de alineamiento requieren ciertos parámetros tales como la matriz de sustitución y el costo por la inserción de *gaps* “-”, los cuales fueron explicados anteriormente.

El tiempo de ejecución cuadrático del algoritmo de Smith-Waterman lo hace impráctico para alineamientos de una secuencias contra bases de datos que contienen

millones de secuencias, cada una con millones de nucleótidos. Para aplicaciones de este tipo se utilizan otras herramientas más rápidas, como BLAST y FASTA, las cuales se basan en algoritmos heurísticos.

BLAST

BLAST [5] (*Basic Local Alignment Search Tool* por sus siglas en inglés), trabaja heurísticamente alineando secuencias de forma local. Fue desarrollado en 1990 por el Centro Nacional para la Información Biotecnológica (*National Center for Biotechnology Information o NCBI*, por sus siglas en inglés) y es ampliamente utilizado para el análisis de secuencias biológicas. BLAST realiza el alineamiento de la secuencia en cuestión (*query sequence*) contra millones de secuencias contenidas en bases de datos, retornando las secuencias en esa base con las cuales se encontró una mayor similitud con la secuencia en cuestión.

El alineamiento con BLAST depende de la base de datos biológicas, los costos de la penalización por insertar *gaps* y la longitud de la palabra inicial (*wordlength*, w), entre otros. BLAST realiza el proceso de alineamiento en tres etapas [5]: siembra (*seeding*), extensión y evaluación.

En la primera etapa, BLAST identifica todos los segmentos de la secuencia en cuestión donde aparecen cada uno de los diferentes fragmentos de longitud w previamente seleccionados. Por ejemplo, suponiendo que la primera secuencia encontrada en la base de datos biológicas es ‘AGAGTAGCC’, si la secuencia en cuestión es ‘CATTAGCG’ y w es igual a 4, entonces el segmento común a las dos secuencias, “la semilla”, sería ‘TAGC’. De esta forma se encuentran todas las secuencias semillas entre la base de datos y la secuencia en cuestión. En la segunda etapa, la búsqueda se extiende hacia ambos lados de cada una de las palabras encontradas en la primera etapa. Durante la extensión se utiliza el algoritmo de Smith-Waterman. En la última

etapa del proceso, BLAST realiza la evaluación estadística de las secuencias semillas extendidas, eliminando alineamientos inconsistentes.

Mediante el proceso anterior, BLAST encuentra los mejores alineamientos llamados pares de alta puntuación (*High Score Pairs o HSPs*, por sus siglas en inglés) [31]. Debido al enfoque heurístico, este método no retorna una solución exacta al problema de optimización [1]. BLAST asigna un significado estadístico a los alineamientos encontrados, utilizando el ***e-value*** (*expected value*); este es un porcentaje relacionado con el número de alineamientos que se espera obtener con un puntaje S o superior, sobre el conjunto de registros presentes en la base de datos biológicas. Cuanto menor sea el valor, más significativo será el alineamiento.

BLAST es un paquete compuesto por diferentes programas: BLASTN, BLASTP, BLASTX, TBLASTN y TBLASTX; la elección del método a utilizar depende básicamente del tipo de la especie biológica a analizar (Tabla 2–6).

Programa	Descripción
BLASTN	Compara una secuencia de nucleótidos desconocida contra una base de datos de nucleótidos.
BLASTP	Compara una secuencia desconocida de aminoácidos contra una base de datos de proteínas.
BLASTX	Compara los marcos de lectura (<i>ORF</i>) de una secuencia de nucleótidos desconocida contra una base de datos de proteínas.
TBLASTN	Compara una secuencia de proteína desconocida contra una base de datos de nucleótidos dinámicamente traducida en los <i>ORFs</i> en ambas direcciones de la traducción.
TBLASTX	Compara los <i>ORFs</i> de una secuencia de nucleótidos teórica desconocida contra los conocidos en una base de datos de nucleótidos.

Tabla 2–6: Tipos de búsqueda ofrecidos por BLAST

Además de realizar los alineamientos, es importante determinar si existe algún tipo de relación biológica asociada a estos. Las distribuciones estadísticas recompi-ladas por Durbin [6] ayudan en esta determinación.

2.4.7. Predicción computacional de genes

Luego de obtener las secuencias de ADN utilizando las técnicas de secuenciación descritas anteriormente, es necesario procesar y analizar las secuencias para extraer información que permite determinar estructura, función y evolución biológica en cada una de las especies. Esta información se encuentra en los genes de la especie.

La búsqueda de los genes es uno de los más grandes desafíos de la Bioinformática. Este proceso comienza con la secuencia de ADN y en la cual se deben encontrar todas las regiones conocidas como “marco abierto de lectura” (*Open reading frame* o *ORF*, por sus siglas en inglés). Como se mencionó anteriormente, en el caso de células eucariotas, los ORF contienen exones e intrones. Computacionalmente, la búsqueda de genes en especies eucariotas es más compleja, debido a la necesidad de distinguir exones de intrones. Las regiones ORF están delimitadas por un codón inicial (ATG) y un codón final (TAA, TAG o TGA). En las secuencias de ADN es posible tener diferentes regiones ORF solapadas, es decir, regiones contenidas unas dentro de otras. En estos casos se tienen en cuenta todas las regiones ORF. Luego de identificar todas las regiones se procede con la búsqueda de los exones, los cuales semánticamente satisfacen alguna de las siguientes reglas:

- Inicio: ATG y fin: GT (*acceptor site*)
- Inicio: AG (*donor site*) y fin: GT
- Inicio: AG y fin: $\{TAA \mid TAG \mid TGA\}$

Con la identificación de los exones se construye el grafo ORF (*ORF graph*), en el cual los nodos representan cada uno de los puntos de inicio y fin de cada exón, y los arcos están definidos por el puntaje obtenido al realizar el alineamiento entre cada exón y la secuencia. Adicionalmente, en cada nodo se adiciona la posición (el índice) en la cual ocurrió la coincidencia en el alineamiento, con respecto a la secuencia inicial de ADN (Figura 2-14).

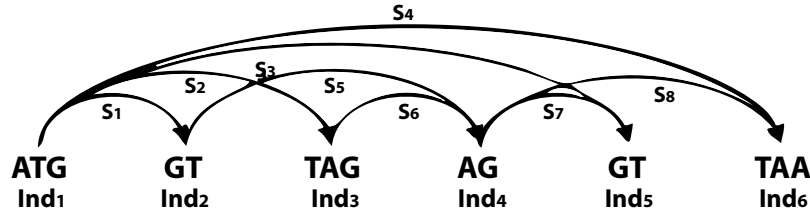


Figura 2–16: Ejemplo del grafo ORF

Posterior a la construcción del grafo, se deben buscar los caminos de mayor peso entre cada codón inicial y final. Para la búsqueda de éstos caminos se utiliza al algoritmo de Dijkstra [32]. Este algoritmo encuentra las regiones que componen el camino de mayor peso entre los dos nodos *source* y *sink* (codón inicial y final). Todas las regiones de ADN que componen cada uno de estos caminos son los posibles genes relacionados con la secuencia biológica. Este método es determinístico y por lo tanto, se distingue de los que usualmente se usan en la búsqueda de genes.

Algoritmo de Dijkstra

También es conocido como el algoritmo de caminos mínimos, fue propuesto en 1959 por Edsger Dijkstra. Permite encontrar el camino más corto entre un nodo inicial y los demás nodos del grafo, a partir del peso de los arcos. La búsqueda de los caminos se realiza sobre la matriz de adyacencia asociada al grafo. Al igual que el algoritmo anterior, es necesario hacer modificaciones al algoritmo para encontrar el camino de mayor peso entre cualquier par de nodos; estos últimos corresponden al codón inicial y final de cada región *ORF*.

Actualmente, existen herramientas computacionales basadas en diferentes enfoques, con capacidad de predecir genes y otras estructuras, algunas de ellas se explican a continuación.

Modelo oculto de Markov

Este modelo, es conocido como HMM por sus siglas en inglés (*Hidden Markov Model*) y representa las transiciones en procesos secuenciales determinadas por una probabilidad de ocurrencia. Utilizando un modelo simple es posible determinar la probabilidad del siguiente nucleótido en la secuencia de ADN, a partir de la cantidad de ocurrencias previas de cada nucleótido. Cada estado tiene asociada una función de probabilidad relacionada con un evento. Estos eventos, por lo general, están relacionados con observaciones de casos conocidos. La figura 2-17 representa un modelo de Márkov con un solo estado aplicado a secuencias de ADN. En este modelo la probabilidad de tener la secuencia ‘AATTATA’ es de $(0,3)^4 * (0,5)^3 = 0,0010$, de forma similar se hace el cálculo para cualquier secuencia de nucleótidos.



Figura 2-17: Modelo de Márkov con un estado

Los parámetros del HMM son determinados a partir de una muestra de secuencias conocidas, con las cuales se realiza el entrenamiento del modelo, para luego realizar la búsqueda de genes en secuencias desconocidas. La prueba y el entrenamiento del modelo se realiza utilizando el algoritmo de Viterbi [33]. Básicamente, este algoritmo construye una estructura de datos llamada *trellis* [34] relacionada con el modelo, la cual requiere $O(ne)$ de espacio, donde n es la longitud de la secuencia y e es el número de arcos en el mejor camino encontrado en la estructura [35].

HMMgene [36] es una herramienta basada en el modelo oculto de Markov. El entrenamiento se realiza a través del criterio llamado “máxima verosimilitud condicional” (*conditional maximum likelihood*) el cual permite maximizar la probabilidad y así realizar la predicción más acertada [8]. Genscan [37], tiene la capacidad de

predecir múltiples genes presentes en la secuencia de ADN; también identifica los intrones. Esta herramienta genera resultados precisos y fiables. Sanja [8] encontró que HMMgene y Genscan fueron las dos herramientas que generaron mayor precisión en la predicción de genes para diferentes secuencias. En estas pruebas se compararon siete herramientas diferentes.

Programación dinámica

Los algoritmos basados en programación dinámica reducen el problema a subproblemas, de los cuales se obtiene la solución del problema total. La programación dinámica resuelve subproblemas una sola vez, y guarda la solución para su futura utilización en la solución de este. El paquete FGENES [38] fue diseñado bajo este principio. FGENES permite encontrar la combinación óptima de todos los exones. El método además utiliza algoritmos de reconocimiento de patrones para construir el conjunto con los posibles genes. Esta herramienta fue entrenada con 600 secuencias de ADN no redundantes relacionadas con especies humanas [8]. Gelfand y Roytgerg [39], diseñaron una herramienta para predecir intrones y exones bajo el enfoque de programación dinámica, usando la estructura de datos llamada *vector dynamic programming*, la cual combina múltiples índices de calidad en la predicción de las estructuras biológicas [11]. GAZE [40], es otra herramienta desarrollada en el estilo de programación dinámica. Es altamente flexible para determinar genes en secuencias de mayor escala de forma eficiente. GAPIII [41] es una combinación de programación dinámica y algoritmos heurísticos que permite predecir genes de forma óptima [35, 42].

Aprendizaje automático

Esta técnica es conocida en inglés como *Machine Learning*. De acuerdo con Arthur Samuel (1959), *Machine Learning* es la capacidad que tienen las computadoras de “aprender” sin ser programadas explícitamente. El aprendizaje o conocimiento es obtenido a través de dos tipos de algoritmos: supervisados y no supervisados. El

aprendizaje supervisado requiere del conjunto de datos de entrenamiento (*training set*) y los resultados deseados para ese conjunto. El conocimiento es obtenido a través de los casos allí presentes y en base a estos casos, el modelo está en capacidad de procesar cualquier otro conjunto de datos. Este enfoque tiene un conocimiento a priori, a diferencia del aprendizaje no supervisado, en el cual se tiene únicamente el conjunto de entrenamiento y los resultados deben ser obtenidos a través de observaciones en el comportamiento de los datos. La precisión del modelo de predicción depende del nivel de similitud entre los datos de entrenamiento y los datos reales. Además existen otros factores que pueden alterar los resultados como los valores extremos (*outliers*). A continuación algunas técnicas y herramientas bajo el enfoque *Machine Learning* desarrolladas para la predicción de genes en secuencias de ADN [43].

Redes bayesianas

- Las redes bayesianas (*Bayesian Networks*) utilizan modelos probabilísticos para representar un conjunto de variables y las relaciones dependientes entre estas. La representación es realizada utilizando un grafo dirigido, donde los nodos están determinados por las variables y los arcos por el valor probabilístico entre cada par de variables dependientes. Las redes bayesianas pueden inferir, estimando las probabilidades posteriores a partir de las variables conocidas. Ya que la representación gráfica permite una mejor visualización de los datos al momento de realizar el respectivo análisis [35]. En la referencia [44] se presenta un modelo bayesiano para la detección de genes, permitiendo introducir al modelo las dependencias estadísticas definidas por expertos.

Redes neuronales artificiales

- Las redes neuronales artificiales (*Artificial Neural Networks o ANN*, por sus siglas en inglés) es un sistema para el procesamiento de información, cuyo comportamiento es similar a las redes neuronales biológicas. El elemento más simple se conoce

como neurona y allí se procesa la información del sistema. Cada neurona recibe un conjunto de datos de entrada y con base en la función de activación genera los diferentes resultados, las neuronas comparten información a través de conexiones donde cada una tiene un peso asociado, por lo general este valor multiplica la señal transmitida [45]. Las redes neuronales son entrenadas a partir de un conjunto de datos de entrenamiento. En muchos casos es necesario realizar miles de repeticiones durante el entrenamiento logrando reducir el margen de error en cada iteración [46]. GRAIL es una herramienta de las más utilizadas y ha permitido encontrar genes para diferentes enfermedades. En una prueba reciente, sobre 110 secuencias compuestas de 829 exones, 134,814 regiones codificantes y 1,257,631 regiones no codificantes; GRAIL reconoció más del 90 % de las bases codificantes con aproximadamente un 5 % de falsos positivos [47].

Árboles de decisión

. Los árboles de decisión son una de las técnicas más utilizadas para representar, clasificar y analizar decisiones secuenciales. Los árboles están basados en el uso de resultados y probabilidades asociadas para la resolución de problemas. En la ejecución del árbol de decisión, se sigue solamente un camino dependiendo del valor de la variable que será evaluada.

GenBlastDT [48] es una herramienta basada en árboles de decisión. Permite encontrar genes en secuencias de proteínas, trabajando de forma rápida y precisa. MORGAN [49] funciona bajo el mismo principio, encontrando los genes en especies vertebradas; además utiliza HMM para identificar las posibles regiones genéticas. Por último, Glimmer [50] determina los genes en especies eucariotas, utilizando funciones de puntaje que determinan la probabilidad de codificar información genética en cada uno de los diferentes fragmentos.

Algoritmos genéticos

. Son algoritmos inspirados en el principio de evolución biológica y selección natural propuesto por Darwin, permiten la resolución de problemas específicos. En la resolución de problemas se utilizan métodos basados en probabilidades. Básicamente estos algoritmos encuentran la solución en tres pasos: selección, recombinación (*crossover*) y mutación, encontrando buenas soluciones hasta alcanzar el resultado óptimo. Saetron [51] desarrolló una herramienta basada en algoritmos genéticos para predecir regiones no-codificantes (*non-coding*) en la especie *Escherichia Coli*, incorporando un nuevo método que utiliza detección automática de patrones para predicción de genes.

Las herramientas mencionadas anteriormente son algunas de las existentes para predecir genes y están basadas en diferentes enfoques computacionales. Luego del proceso de identificación de los genes, esta información es analizada por expertos (biólogos, genetistas, entre otros), realizando análisis filogenético entre las especies, determinando el comportamiento de las redes de regulación génica al interior de las especies, estudiando genotipos y *pathways*, entre otros análisis biológicos [35].

Capítulo 3

IMPLEMENTACIÓN DEL SISTEMA

El sistema permite aplicar diferentes métodos computacionales para el análisis de secuencias de ADN. En este capítulo se presenta más información sobre la especie que fue caso de estudio de esta investigación, además la descripción detallada cada uno de los métodos del sistema.

3.1. Datos experimentales

El problema de la identificación de los genes se usó como caso de estudio para la elaboración del sistema, secuencias de ADN relacionadas con la levadura *Debaryomyces Hansenii*. De acuerdo con el NCBI (*National Center for Biotechnology Information*) y el EBI (*European Bioinformatics Institute o EBI*) esta levadura fue secuenciada en siete cromosomas diferentes. Se recibieron las secuencias de ADN de seis cepas de una variante de esta levadura. Cada una de las secuencias se encontraba en formato FASTQ, y cada uno de estos archivos fueron generados por laboratorios a partir de muestras biológicas de la especie. Estas muestras se procesaron utilizando la técnica de secuenciación descrita anteriormente HiSeq (*High Throughput Sequencing*) [20].

Para cada una de las cepas se tienen dos archivos FASTQ los cuales deben ser unificados como un primer paso para encontrar la secuencia de ADN asociada a cada

cepa. La información contenida en los dos archivos es requerida para luego tener una aproximación a la secuencia original de cada cepa. El ensamble de los archivos se realiza a través del ensamblador VELVET, el cual tiene la capacidad de procesar el formato FASTQ Solexa/Illumina. VELVET [7] fue elegido debido a los resultados obtenidos en comparaciones realizadas por la compañía Illumina [28]. VELVET a partir de los FASTQ genera un archivo en formato multi-fasta. Este archivo contiene miles de secuencias cortas de ADN llamadas *contigs*.

A partir de estos *contigs* comienza el proceso de reconstrucción de la secuencia de ADN, esto en cada una de las cepas entre los siete cromosomas de la levadura *D. Hansenii*. Los *contigs* son los datos de entrada del sistema desarrollado en esta investigación y sobre los cuales se comienzan a aplicar cada uno de los métodos implementados.

3.2. Métodos implementados

Para el desarrollo del sistema se utilizó el lenguaje de programación *Python*, debido a la claridad en su sintaxis, la portabilidad y el amplio compendio de librerías con que este cuenta. Durante el desarrollo del sistema se trabajó en un servidor Linux *Intel XEON*, el cual tiene 2 Procesadores *Quad Core* y 48Gb en memoria RAM. Básicamente, el sistema tiene cinco métodos que permiten realizar diferentes análisis a partir de secuencias de ADN, a saber:

- Método para elegir los costos de penalización (*open* y *extended gap penalty*). Permite apoyar la decisión al momento de elegir los valores para d y e , requeridos en el alineamiento.
- Método para validar el significado de los alineamientos, permite evaluar si los alineamientos obtenidos fueron debido a coincidencia o por el contrario podrían tener algún significado biológico.

- Método para clasificar los *contigs* generados por VELVET. Permite la clasificación realizando los dos tipos de alineamientos: BLAST y Smith-Waterman.
- Método para reconstruir una sola secuencia de ADN por cada cromosoma a partir de los *contigs* clasificados previamente.
- Método que permite buscar exhaustivamente las posibles regiones que codifican información genética a partir de las secuencias de ADN.

A continuación se explica con más detalle cada uno de los métodos mencionados anteriormente y todo lo relacionado con el procesamiento de los *contigs*, utilizando los datos de la especie *D. Hansenii*.

3.2.1. Alineamiento de secuencias

El sistema permite realizar el alineamiento de secuencias bajo dos enfoques: algoritmo de Smith-Waterman y BLAST. Para la ejecución de cada uno se utilizan algunas librerías disponibles en Python, al igual que el paquete para bioinformática *Biopython* (más información en <http://www.biopython.org/wiki/Biopython>). A continuación se explica la forma en que se implementó cada uno de los dos enfoques anteriores.

Algoritmo de Smith-Waterman

El alineamiento local se realiza utilizando la función *water* encontrada en el paquete *EMBOSS* [52]. Este paquete es provisto por la Organización Europea de Biología Molecular (*The European Molecular Biology Open Software Suite* o *EMBOSS*, por sus siglas en inglés) y está disponible para ser utilizada desde Python. El algoritmo *water*, fue optimizado con el fin de mejorar la velocidad de ejecución. Este algoritmo permite realizar el alineamiento local en dos secuencias de ADN, con la opción de parametrizar los costos del *open* y el *extended gap*, al igual que la matriz de sustitución, retornando el alineamiento óptimo entre las dos secuencias (Ver Figura 3-1).

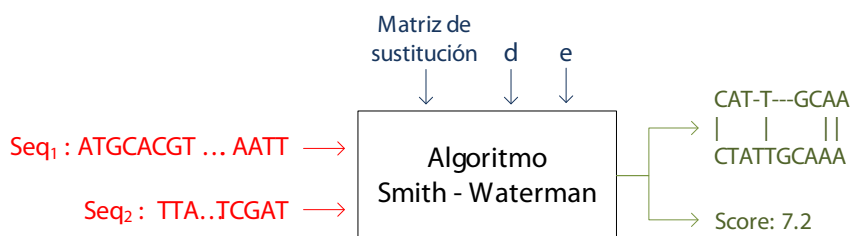


Figura 3-1: Estructura de la función *water*

Para realizar el alineamiento de los miles de *contigs*, la implementación de la función *water* se realizó en paralelo, utilizando la librería *Parallel Python* (más información en <http://www.parallelpython.com>). Esta librería proporciona un mecanismo para la ejecución paralela de código Python en sistemas *multi-core* o *clusters*, resultando una paralelización similar a la de las librerías *OpenMP* o *MPI*. Bajo este principio, la función *water* se diseñó para ser ejecutada concurrentemente en cada uno de los *threads* disponibles del servidor. Cada uno de estos trabaja bajo la supervisión de un *thread* maestro, el cual se encarga de consolidar la información retornada por cada *thread*. La figura 3-2 representa el modelo sobre el cual se ejecuta el algoritmo de Smith-Waterman, la cantidad de *cores* es parametrizable y depende de la disponibilidad de recursos en el servidor.

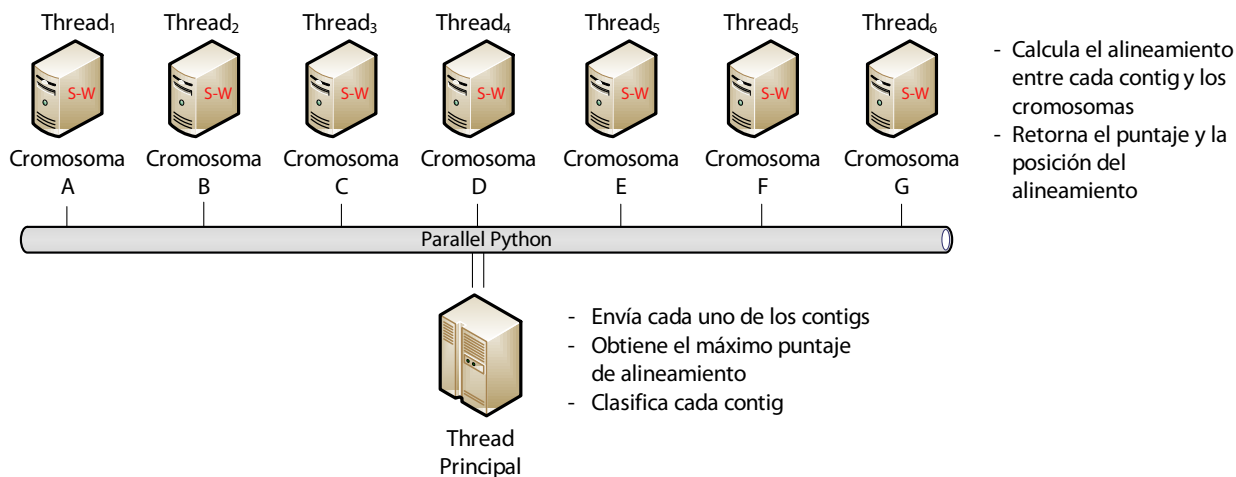


Figura 3-2: Modelo de ejecución del algoritmo Smith-Waterman

Bajo este diseño, se reduce el tiempo de ejecución del algoritmo Smith-Waterman, pues la paralelización de este algoritmo sigue siendo un gran desafío para los investigadores en computación de alto rendimiento (*High Performance Computing o HPC* por sus siglas en inglés). En la actualidad, existen diferentes versiones que intentan reducir el tiempo computacional del algoritmo [53], algunas están basadas en el uso de FPGAs (*Field Programmable Gate Arrays*) [54]. Existe una versión modificada del algoritmo Smith-Waterman que utiliza solamente $O(m + n)$ de espacio, pero aumenta el tiempo de ejecución. Esta versión fue propuesta en el algoritmo de Hirschberg, diseñado para resolver el problema de la subcadena común más larga [55].

Alineamiento con BLAST

El Centro Nacional para la Información Biotecnológica (*NCBI*), ofrece un paquete de software en el cual se encuentra implementado BLAST [56]. Este paquete está desarrollado en lenguaje C++ y tiene la capacidad de trabajar de forma local, lo cual permite disminuir tiempos, al no tener que comunicarse con los servidores de NCBI. Para realizar el alineamiento de secuencias fue necesario descargar la base de datos biológicas “nt/nr” (*Nucleotide Collection*), en la cual están contenidas las secuencias de ADN de los siete cromosomas de la especie *D. Hansenii*.

Para el alineamiento de secuencias, el paquete requiere como datos de entrada la secuencia en cuestión, el nombre de la base de datos biológicas y el valor de w , el cual determina el tamaño de las palabras “semilla”. También es posible parametrizar otros valores como el valor del umbral (*e-value*); este permite reducir el espacio de búsqueda al momento de realizar el alineamiento. BLAST retorna todos los mejores alineamientos, llamados pares de alta puntuación (*High Score Pairs o HSPs*) (Ver Figura 3-3).

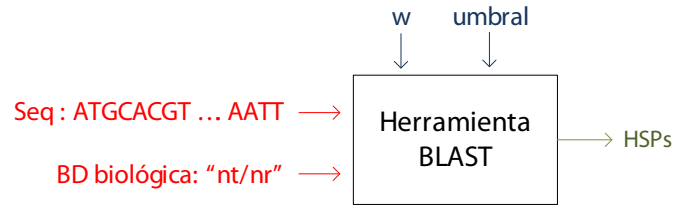


Figura 3-3: Estructura de la herramienta BLAST

Durante el desarrollo del sistema, se diseñó un método en Python que utiliza la herramienta BLAST, permitiendo clasificar los miles de *contigs* de forma más rápida que el algoritmo anterior. Este método no requiere de paralelización, pero es importante recordar que con BLAST se sacrifica precisión en pro de la velocidad de ejecución.

Los dos algoritmos anteriores, BLAST y Smith-Waterman, fueron utilizados para alinear cada uno de los *contigs* contra cada uno de los siete cromosomas. El uso de los dos métodos permite comparar resultados, analizar ventajas y desventajas sobre cada uno, y posiblemente en algunos casos, como se describe más adelante, combinar los resultados para tomar decisiones sobre la clasificación de los *contigs*.

3.2.2. Elección de los costos de penalización

Como se indicó anteriormente, para el alineamiento de secuencias se requiere de la matriz de sustitución y los costos de penalización por insertar *gaps* durante el alineamiento. Estos costos, d (*open gap*) y e (*extended gap*), no pueden ser elegidos de forma arbitraria, debido al cambio significativo en los resultados. Por esta razón, se propone un método probabilístico para determinar estos valores, el cual está basado en la teoría probabilística propuesta por Durbin [6].

Retomando la ecuación que determina el costo afín (3.1), donde g es la longitud del *gap*, y en base a la relación propuesta por Durbin (3.2), en este trabajo se supone

que $f(g)$ es la función identidad, es decir, $f(g) = g$; en base a lo anterior se obtiene la relación matemática (3.3),

$$\gamma(g) = d + (g - 1)e \quad (3.1)$$

$$\gamma(g) = \log(f(g)) \quad (3.2)$$

$$d + (g - 1)e \approx \log(g) \quad (3.3)$$

A partir de la relación matemática obtenida (3.3), se propone un método heurístico que permite determinar los valores para d y e . Este método es presentado en el pseudocódigo 3.

Pseudocódigo 3 (Determinar los costos de penalización)

Entrada: Las secuencias que serán alineadas: Seq_1 y Seq_2

Salida: Los valores más acordes para d y e

Inicialización:

$umbral \leftarrow 0.05$; $d \leftarrow 0.5$; $e \leftarrow 0.5$; $i \leftarrow 1$; $noIter \leftarrow 15$; $c_error \leftarrow 1$

mientras ($i \leq noIter$ o $c_error > umbral$) **hacer**

$alineamiento \leftarrow SmithWaterman(Seq_1, Seq_2, d, e)$

$gaps \leftarrow longitudGaps(alineamiento)$

$c_error \leftarrow \frac{1}{tam(gaps)} \sum_{g \in gaps} [(d + (g - 1)e) - \log(g)]^2$

$error(i) \leftarrow c_error$

$corte, pend \leftarrow regresionLineal(gaps, \log(gaps))$

$d \leftarrow corte$; $e \leftarrow pend$

$i \leftarrow i + 1$

fin mientras

devolver d, e

En cada una de las iteraciones se utiliza el algoritmo de Smith-Waterman bajo el diseño explicado en la sección anterior. El método iterativo descrito anteriormente, converge a un único valor para d y e . Este algoritmo se aplica a una muestra aleatoria

de *contigs* para cada una de las cepas, obteniendo los valores más acordes para d y e .

3.2.3. Significado biológico de los alineamientos

Luego de alinear las secuencias de ADN y encontrar el mejor alineamiento, es necesario descartar que el alineamiento obtenido no es resultado del azar. Por lo tanto, es importante validar el significado biológico de los alineamientos, recordando que una de las razones por la cuales se comparan las secuencias es para determinar si existe algún tipo de relación biológica entre las especies, y así conjeturar algún tipo de evolución entre las mismas.

De acuerdo con Durbin [6], existen dos métodos que permiten validar el significado biológico del alineamiento. El primero está basado en la construcción y comparación de diferentes modelos utilizando probabilidades Bayesianas. El segundo es el enfoque clásico, basado en la distribución de frecuencias de los puntajes luego de realizar diferentes alineamientos.

El método diseñado en el sistema para validar el alineamiento está basado en el segundo enfoque de los propuestos por Durbin. Este utiliza principios estadísticos para analizar la distribución de frecuencias de los puntajes de los alineamientos. El siguiente pseudocódigo presenta el método desarrollado, el cual permite validar el significado del alineamiento entre las dos secuencias Seq_1 y Seq_2 , cuyas longitudes son m y n respectivamente:

Si el puntaje del alineamiento óptimo entre Seq_1 y Seq_2 , es comparado con una distribución gaussiana de los puntajes de alineamientos aleatorios contra Seq_2 ; el grado de desviación del alineamiento se interpretará como indicativo de la influencia de un factor biológico, descartando que se trata de un resultado obtenido por cuestión del azar.

La figura 3–4 representa la distribución de los puntajes en una muestra tomada como ejemplo, en la cual el punto rojo es el *score* del alineamiento inicial. Además

Pseudocódigo 4 (Validar el significado de los alineamientos)

Entrada: Las secuencias Seq_1 y Seq_2 , y los valores d y e

Salida: Distribución de los puntajes

Inicialización:

$i \leftarrow 1, noIter \leftarrow 200$

$S \leftarrow SmithWaterman(Seq_1, Seq_2, d, e)$

mientras ($i \leq noIter$) **hacer**

$Seq_i \leftarrow seqAleatoria(tamaño(Seq_2))$

$puntajes(i) \leftarrow SmithWaterman(Seq_1, Seq_i, d, e)$

$i \leftarrow i + 1$

fin mientras

$freccs \leftarrow obtenerFrecuencias(puntajes)$

$pvalue \leftarrow calcularPvalue(puntajes)$

$graficarDistribucion(freccs, S, p_value)$

se encuentra el valor del $pvalue$ relacionado con los puntajes, si este es mayor de 0.05, los datos corresponden a una distribución normal. Por el tipo de distribución de los datos y la ubicación del puntaje de alineamiento S , se puede concluir que el alineamiento analizado tiene un alto significado, pues el mismo no corresponde al puntaje de un alineamiento aleatorio [30, 57]. Se espera encontrar muchos alineamientos al azar con puntuaciones bajas, pero pocos alineamientos con puntuaciones altas. Estos resultados son consistentes con modelos empíricos estudiados en [58, 59], demostrando probablemente que el alineamiento está fundamentado en algún tipo de relación biológica.

Luego de determinar los valores acordes para d y e con el método anterior, se valida el significado biológico de los alineamientos para una muestra aleatoria de *contigs*. El alineamiento es realizado entre cada *contig* y cada uno de los cromosomas de la especie. Con base en la distribución de frecuencias es posible validar si los alineamientos obtenidos tienen algún significado biológico. Para tener una mejor precisión en los puntajes, el alineamiento es realizado utilizando el algoritmo de Smith-Waterman.

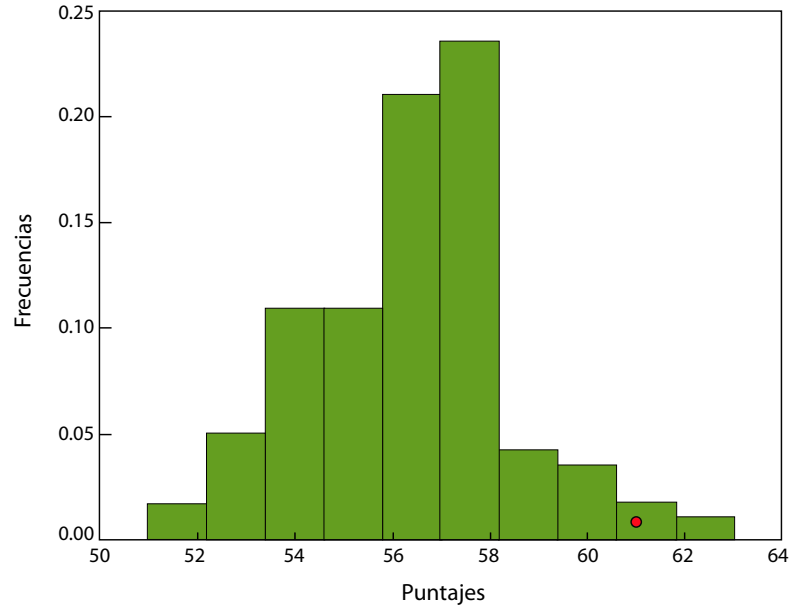


Figura 3–4: Distribución de puntajes para determinar el significado biológico

3.2.4. Clasificación de los *contigs*

Como se mencionó anteriormente, el ensamblaje de los archivos FASTQ produce un archivo en formato multi-FASTA con miles de secuencias de ADN cortas llamadas *contigs*. Para reconstruir una secuencia de ADN a partir de los *contigs* es necesario clasificarlos en cada uno de los siete cromosomas de la especie *D. Hansenii*. Esto se realiza en cada una de las cepas y en dos fases.

En la primera fase, se utiliza el algoritmo de Smith-Waterman para alinear cada uno de los *contigs* contra los siete cromosomas de la especie. El *contig* se asocia al cromosoma cuyo *score* de alineamiento es mayor. Debido a que el mismo puntaje puede aparecer en más de un cromosoma, la clasificación no siempre es completa y se hace necesaria una segunda fase. En esta segunda fase realiza el desempate, utilizando los dos algoritmos (Smith-Waterman y BLAST), para así lograr completar la clasificación.

Puede ocurrir que después de intentar desempatar con BLAST y Smith-Waterman el empate aún persista. De hecho, suponiendo que para el *contig* x , Smith-Waterman

arroje el mayor puntaje con los cromosomas A, B y C, y luego al alinear el mismo *contig* con BLAST se tenga que:

1.) BLAST retorne mejor alineamiento con los cromosomas A y B. En este caso se obtiene como resultado los cromosomas comunes a las dos clasificaciones. Es decir, el *contig* x se clasifica en los cromosomas A y B .

2.) BLAST retorne el mejor alineamiento con los cromosomas D y E. En este caso no existen cromosomas comunes a las dos clasificaciones. Por lo tanto, la información hasta este momento no sería suficiente para completar la clasificación del *contig* x .

En este último caso, se hace la comparación entre el *e-value* retornado por BLAST y el calculado a partir del puntaje obtenido con Smith-Waterman, seleccionando el algoritmo con el cual se obtuvo el menor *e-value*. A continuación se explica el método implementado para este segundo caso.

Clasificación utilizando el *e-value*

Altschul y Karlin [60], definieron la ecuación (3.4) la cual, en base al tamaño del espacio de búsqueda ($m * n$), el parámetro (λ), el puntaje normalizado (S') y una constante (k), permite determinar si el alineamiento fue obtenido por azar. El *e-value* está definido como:

$$e - value = kmn(exp)^{-\lambda S'} \quad (3.4)$$

El espacio de búsqueda está determinado por los valores m y n . El primer valor, m , es la longitud de la secuencia a alinear y n es el tamaño de la base de datos biológicas. Los valores de k y λ están asociados a la matriz de sustitución y a los costos de penalización, y el valor de S' corresponde al puntaje del alineamiento

normalizado. De acuerdo a lo mencionado en la referencia [31], el valor sugerido para k es 0.1; lo cual es un valor despreciable y no genera alteración alguna en los resultados. De igual forma, se utilizó el código en *perl* allí propuesto que permite calcular el valor de λ a partir de los valores de *match* y *mismatch* definidos en la matriz de sustitución *EDNASIMPLE*. El valor del puntaje (S) obtenido con el algoritmo de Smith-Waterman debe ser normalizado, para posteriormente encontrar el *e-value*. La ecuación (3.5), permite encontrar el valor de S' .

$$S' = \frac{\lambda S - \ln(k)}{\ln(2)} \quad (3.5)$$

Con base en las dos ecuaciones anteriores, es posible calcular el *e-value* utilizando el puntaje obtenido con el algoritmo de Smith-Waterman, para luego compararlo con el valor retornado por BLAST. La comparación se realiza para uno de los *contigs* en los que se presenta el empate y se usa para clasificar el *contig* cuyo alineamiento generó el menor *e-value*. De esta manera, se logra resolver el empate, en el caso de que no existen cromosomas comunes a las dos clasificaciones para cada *contig*. La figura 3-5 ilustra con un ejemplo el método de clasificación a partir del *e-value*.

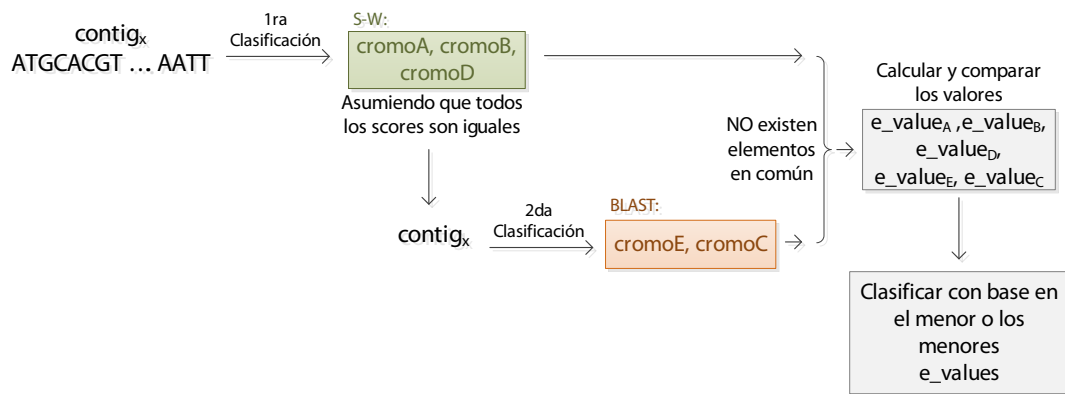


Figura 3-5: Clasificación de los *contigs* a partir del *e-value*

La figura 3-6 describe de manera sencilla el proceso completo de clasificación.

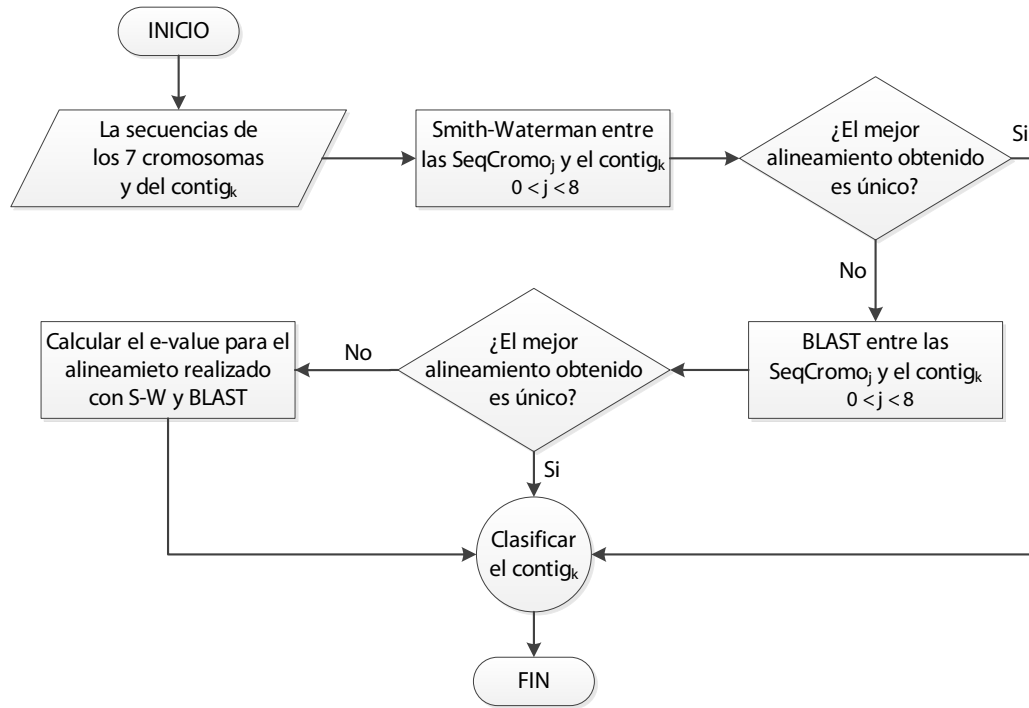


Figura 3–6: Representación del proceso de clasificación

Durante el proceso de alineamiento se utilizan los costos de penalización d y e determinados con el segundo método. Para el alineamiento con Smith-Waterman se utilizó la matriz de sustitución *EDNASIMPLE* y la base de datos biológicas utilizada para el alineamiento con BLAST fue “nt/nr”.

3.2.5. Reconstrucción de la secuencia

Luego de clasificar los *contigs* en cada uno de los cromosomas, se hace necesario realizar un proceso de unificación de estos. Con el fin de lograr tener una sola secuencia que se aproxime a la secuencia de ADN para cada cromosoma, se implementó un método, que permite reconstruir la secuencia de ADN a partir de los *contigs* clasificados, obteniendo una sola secuencia de ADN para cada cromosoma. La figura 3–7 representa de manera simple la idea principal de este método. El valor de x corresponde a cada uno de los siete cromosomas de la especie *D. Hansenii*.

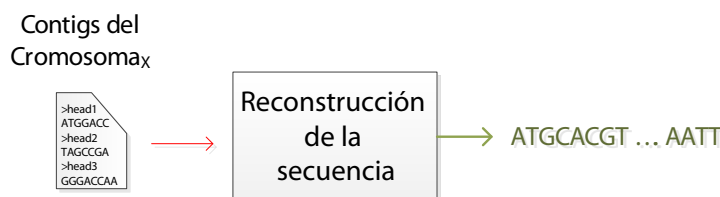


Figura 3–7: Proceso de reconstrucción realizado en cada cromosoma

Al momento de realizar el alineamiento entre cada *contig* y cada cromosoma, además del puntaje, se guarda la posición del cromosoma en la cual se encontró el mejor alineamiento. Esta posición está compuesta por el índice superior e inferior, así como se ilustra en la figura 3–8.



Figura 3–8: Índices obtenidos en el alineamiento

A partir de esta posición se comienza el proceso de unificación de todos los *contigs*. Durante este se pueden presentar tres posibles casos entre cada par de *contigs*:

- 1.) **Los *contigs* son contiguos.** Este es el caso ideal, durante el proceso de reconstrucción de la secuencia; simplemente se realiza el empalme entre las dos secuencias de los *contigs*.
- 2.) **Los *contigs* son disjuntos.** En este caso se encuentran separados por espacios en blanco. Para completar estos espacios se extrae de la secuencia original del cromosoma el fragmento de secuencia faltante, teniendo en cuenta la posición donde ocurrió el espacio en blanco.
- 3.) **Los *contigs* están superpuestos.** En este caso se encuentran solapados, es decir en una misma posición pueden existir más de un nucleótido, por lo tanto se hace necesario aplicar un consenso entre estos nucleótidos y así determinar el

valor final para dichas posiciones.

Los tres posibles casos se representan en la figura 3–9. En el último caso es necesario tener presente el consenso entre las regiones de los *contigs* solapadas. Para esto se diseñó un método que permite resolver este solapamiento y el cual se explica a continuación.

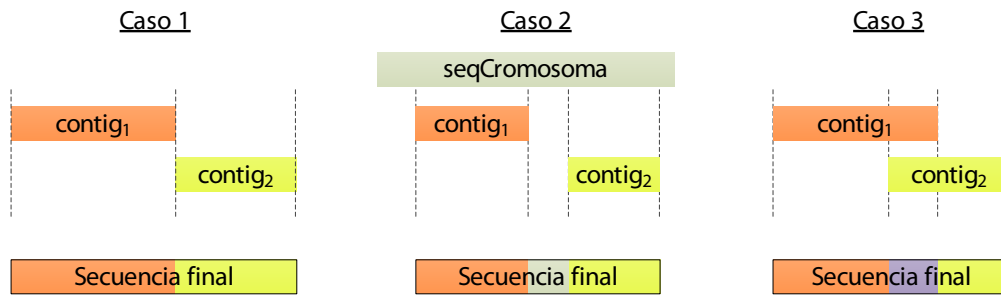


Figura 3–9: Casos presentes durante la reconstrucción de la secuencia

Consenso entre los nucleótidos superpuestos

Al momento de reconstruir la secuencia se pueden presentar *contigs* superpuestos, por lo cual se propone un procedimiento que permite determinar el valor final en dichas posiciones de la secuencia. Este procedimiento, permite resolver el escenario planteado en el caso 3. En cada una de las posiciones que se presenta el solapamiento, se calcula la cantidad de veces que aparece cada nucleótido (A, T, C o G) teniendo presente todos los *contigs* que generan el solapamiento. Se elige el nucleótido que obtuvo el mayor número de ocurrencias. El procedimiento diseñado tiene un beneficio adicional, y es que permite parametrizar un valor de umbral, el cual define qué tan superior es una cantidad con respecto a otra. Por ejemplo, suponiendo que se tienen 20 *contigs* solapados y luego de hacer el conteo múltiple en la posición i , se obtiene: A:10, T:9 y C:1. En este caso, no está completamente justificado el concluir que el nucleótido A debe ir en la posición i de la secuencia final, debido a la mínima diferencia que existe con la cantidad de T. Este método permite

definir un valor de umbral, para determinar cuándo una cantidad se considera muy superior a otra, el cual se puede parametrizar en términos porcentuales.

En la figura 3–10 se presenta un ejemplo, en el cual se resuelve el solapamiento entre cinco *contigs* con valor de umbral cero. Cuando existe un empate en la cantidad de ocurrencias, el método inserta el valor *N* para indicar que cualquiera de los cuatro nucleótidos podría estar presente en dicha posición.

contig	Sequences																
1	A	T	C	A	C	A	C	C	T	C	T	T	A	A	T	G	C
2								A	T	G	G	A	T				
3					C	G	T	C	T	C	T	A	A	A			
4										G	G	A	T	T	A		
5					C	A	C	A	C	C	C	A	T	T	A		
	A	T	C	A	C	A	C	N	T	C	N	A	T	N	A	G	C

Figura 3–10: Ejemplo del método consenso

3.2.6. Predicción de genes

Luego de la construcción completa de la secuencia de ADN a partir de los *contigs* en cada cromosoma, se realiza la búsqueda de aquellas regiones que contienen información genética, eso es, los posibles genes en la secuencia. Para esto, se comienza con buscar las regiones ORF en la secuencia, las cuales están delimitadas por un codón inicial (ATG) y un codón final (TAA, TAG o TGA). Es posible encontrar en la secuencia diferentes regiones ORF solapadas unas con otras. En la figura 3–11, las regiones en color amarillo representan los codones iniciales y las de color rojo los codones finales presentes en una secuencia de ejemplo. La búsqueda de estas regiones se realiza de forma exhaustiva a lo largo de toda la secuencia de ADN.

Luego de identificar todas las regiones ORF en la secuencia se procede con la búsqueda de los exones. Los exones satisfacen las siguientes reglas:

- Inicio: ATG y fin: GT



Figura 3–11: Ejemplo de solapamiento en las regiones ORFs

- Inicio: AG y fin: GT
- Inicio: AG y fin: $\{TAA \mid TAG \mid TGA\}$

Con base en las reglas anteriores, en la figura 3–12 se representan con tres colores los posibles exones presentes en una región ORF dada como ejemplo. Al igual que las regiones ORF, los exones también se pueden encontrar superpuestos. Luego de detectar todos los exones se crea una estructura de datos, conocida como el grafo ORF.

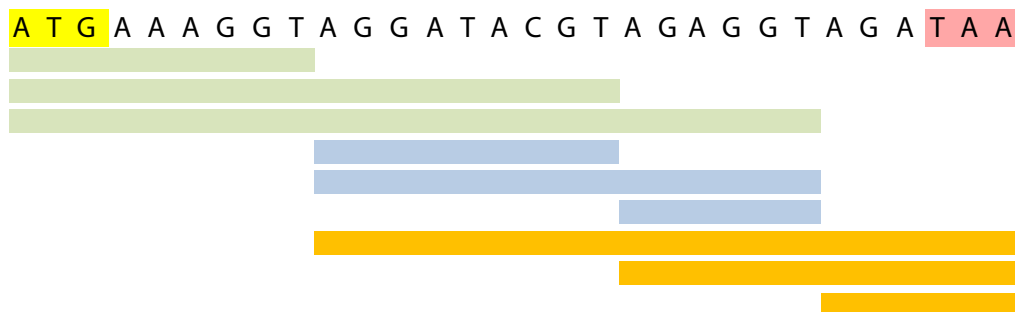


Figura 3–12: Ejemplo de exones en una región ORF

En el grafo ORF los nodos representan cada una de las regiones comienzo y fin de cada exon, y el peso de los arcos está determinado por el valor obtenido al alinear el exon contra la secuencia completa del cromosoma. Este alineamiento se realiza utilizando el algoritmo de Smith-Waterman o BLAST, según sea requerido. La construcción del grafo ORF se realiza en Python utilizando la librería *networkx*, (más información en <http://www.networkx.lanl.gov/>). Entre cada par de nodos se guarda la secuencia completa del exon y la posición del cromosoma en la cual se obtuvo el mejor alineamiento local.

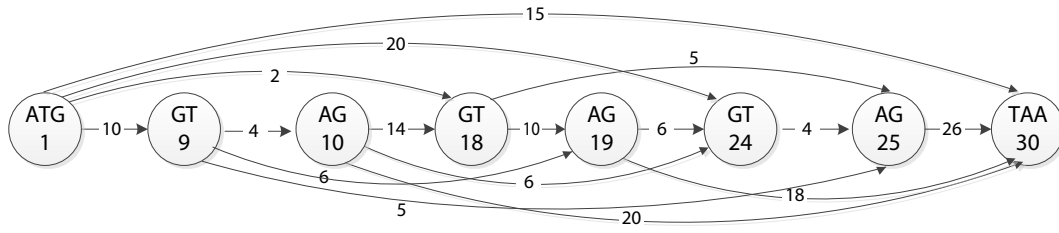


Figura 3–13: Ejemplo del grafo ORF

La figura 3–13 ilustra el grafo ORF obtenido para la secuencia ejemplo de la gráfica anterior. En este se estructuran todos los exones contenidos en la secuencia. Además de crear el grafo, también es posible generar un archivo con la información contenida en el grafo, así como se presenta en la figura (3–14). Para los exones que contienen la letra *N* asignada con el método de consenso, se realiza el alineamiento sin tener presente este nucleótido, debido a la alteración que se puede presentar en el resultado del alineamiento.

```

ATG-1, TAA-30 {weight=15, idx='1:30', sequence='ATGAAAGGTAGGATACGTAGAGGTAGATAA'}
ATG-1, GT-24 {weight=20, idx='1:24', sequence='ATGAAAGGTAGGATACGTAGAGGT'}
ATG-1, GT-18 {weight=2, idx='1:18', sequence='ATGAAAGGTAGGATACGT'}
ATG-1, GT-9 {weight=10, idx='1:9', sequence='ATGAAAGGT'}
AG-10, GT-30 {weight=20, idx='10:30', sequence='AGGATACGTAGAGGTAGATAA'}
AG-10, GT-24 {weight=6, idx='10:24', sequence='AGGATACGTAGAGGT'}
AG-10, GT-18 {weight=14, idx='10:18', sequence='AGGATACGT'}
AG-19, GT-30 {weight=18, idx='19:30', sequence='AGAGGTAGATAA'}
AG-19, GT-24 {weight=6, idx='19:24', sequence='AGAGGT'}
AG-25, GT-30 {weight=23, idx='25:30', sequence='AGATAA'}

```

Figura 3–14: Información contenida en el grafo ORF

Luego de la construcción del grafo ORF, se procede con la búsqueda de los caminos de mayor peso entre cada codón inicial y final de la región ORF. Para esto se implementaron algoritmos basados en el algoritmo de Dijkstra. La implementación está inspirada en la solución de Edsger Dijkstra, y permite encontrar el camino más costoso entre los dos nodos *source* y *sink* (codón inicial y final). La unión de los fragmentos de secuencia que forman parte del camino más costoso dan origen a una nueva secuencia. Estos caminos tienen alta probabilidad de corresponder a regiones

codificantes. Esta secuencia puede contener una o más letras N , las cuales fueron insertadas en el método consenso.

El procedimiento anterior, permite encontrar todos los posibles genes presentes en una región ORF, por lo cual se vuelve iterativo para cada una de las regiones ORF encontradas previamente en la secuencia. Todos los grafos ORFs son desconexos y deben ser analizados de forma independiente. Para reducir la cantidad de regiones ORF en la secuencia, se ha introducido un valor de umbral. Este valor permite buscar únicamente las regiones ORF donde la longitud es inferior a este umbral.

El método anterior es aplicado a cada una de las regiones ORF encontradas previamente. Este realiza la búsqueda de los genes de forma exhaustiva a lo largo de toda la secuencia, dejando de lado el uso de métodos estadísticos, analizando todas las regiones de la secuencia. Al final de la búsqueda se obtiene el conjunto de todas las secuencias que generaron los caminos más costosos, las cuales forman parte del posible conjunto de genes.

Evaluación probabilística

Para cada uno de los posibles genes encontrados se realiza un último procedimiento, encontrar un valor probabilístico para cada una de las secuencias. Este valor permite determinar la probabilidad de que cada secuencia sea un gen, esto es, en base a los genes actuales de la especie *D. Hansenii*. A cada uno de los posibles genes se le aplica lo descrito en el siguiente pseudocódigo 5.

Este método fue desarrollado para cumplir básicamente con dos objetivos. Primero, permite filtrar aquellas secuencias las cuales no arrojan alineamiento con ninguno de los genes de la especie *D. Hansenii*. Y en segundo lugar, permite asignar una probabilidad a la secuencia gen que fue predicha. Para esto se tienen en cuenta la cantidad de genes con los cuales BLAST encontró algún tipo de alineamiento, sobre la cantidad total de alineamientos reportados. También se informa: el puntaje

Pseudocódigo 5 (Determinar probabilidad de ocurrencia de una secuencia gen)

Entrada: Las secuencias de las posibles regiones codificantes: *Regiones*
Salida: Probabilidad de ocurrencia y el mayor puntaje de los alineamientos

```

para todo gen  $\in$  Regiones hacer
    lista_alineamientos  $\leftarrow$  BLAST(gen, nt/nr)
    lista_genes  $\leftarrow$  obtenerGenes(lista_alineamientos)
    probabilidad  $\leftarrow$  tamaño(lista_genes)/tamaño(lista_alineamientos)
    max_score  $\leftarrow$  obtenerMaxScore(lista_genes)
    prom_score  $\leftarrow$  promScores(lista_genes)
    imprimir Máximo Score: max_score
    imprimir Promedio: prom_score
    imprimir Probabilidad: probabilidad
fin para
  
```

de alineamiento promedio y el puntaje máximo obtenido por dicha secuencia. Esto le permite al sistema tener algo más de exactitud sobre el conjunto de genes predichos. De forma similar, se reporta la posición en la secuencia en la cual se encontró el posible gen.

Capítulo 4

ANÁLISIS DE RESULTADOS

En este capítulo se muestran los resultados obtenidos luego de utilizar una propuesta multialgorítmica para el análisis de secuencias de ADN de especímenes aparentemente relacionadas a la levadura *Debaryomyces Hansenii*. Los resultados presentados están relacionados con las secuencias asociadas a la primera y segunda cepa de la nueva especie. La figura 4–1 presenta las etapas del análisis de las secuencias. Como se indicó anteriormente, los archivos de entrada del sistema están en formato multi-FASTA y contienen miles de *contigs* para cada una de las cepas de la especie.

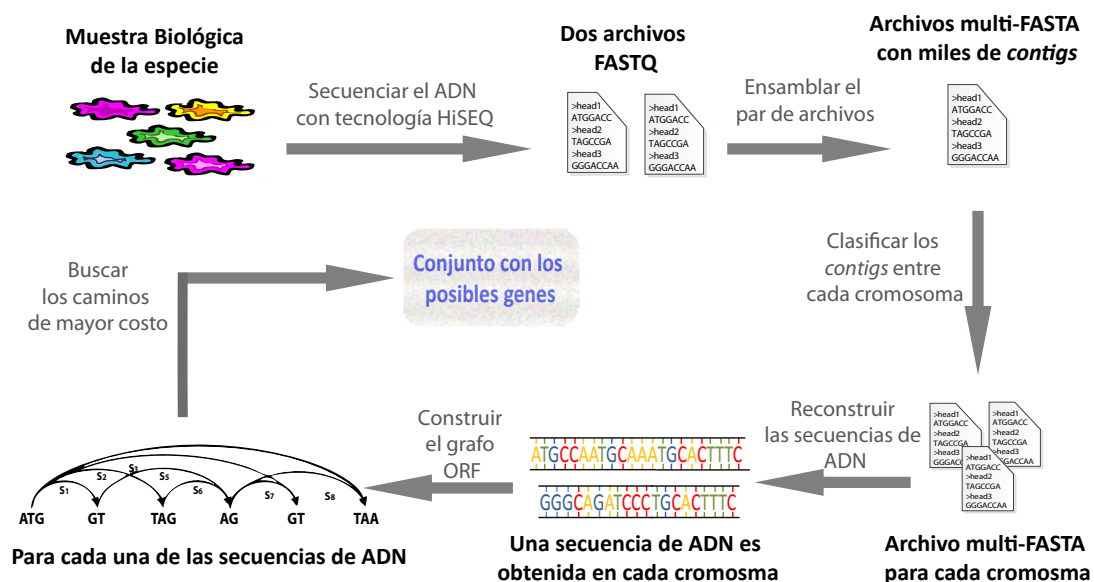


Figura 4–1: Procesos en el análisis de secuencias y búsqueda de genes

4.1. Ensamble de los archivos

Las secuencias de ADN en formato FASTQ fueron generadas por secuenciadores de segunda generación, basados en la tecnología HiSEQ (*High Throughput Sequencing*). Cada una de las cepas produjo dos archivos complementarios FASTQ, en los cuales se representan los nucleótidos de la secuencia de ADN.

Durante esta investigación se trabajó únicamente con los archivos relacionados a dos cepas. El tamaño del archivo de la primera cepa es 4.2Gb, mientras que el de la segunda, es 3.9Gb. Cada par de archivos en FASTQ fué ensamblado con VELVET. Como resultado de este ensamblaje se obtuvo un archivo en formato multi-FASTA, con 32,510 *contigs* de longitud promedio 135 para la primera cepa. Para la segunda cepa, 33,665 *contigs* con una longitud promedio de 136 (Tabla 4–1).

Número de la Cepa	Id. de la Cepa	Número de <i>contigs</i>	Longitud promedio
1	J26	32,510	135
2	398	33,665	136

Tabla 4–1: Información sobre los *contigs* obtenidos para las dos primeras cepas

4.2. Clasificación de los *contigs*

La expresión “clasificación de los *contigs*”, se refiere al proceso de alinear los *contigs* contra las secuencias de los cromosomas de la especie *D. Hasenni* ya conocidos. Previo a esta clasificación fue necesario elegir las metodologías de alineamiento y selección de parámetros. Entre las cuales están:

- 1.) Determinar los costos de penalización (d y e) apropiados para realizar el alineamiento.
- 2.) Encontrar los valores de d y e más apropiados, en base al significado biológico de una muestra aleatoria de alineamientos.
- 3.) Realizar el primer alineamiento utilizando el algoritmo de Smith-Waterman.

4.) Aplicar el algoritmo de BLAST como mecanismo de refinamiento y resolución de ambigüedades de clasificación para los *contigs*.

A continuación se discuten los resultados obtenidos al aplicar esta metodología multialgorítmica:

4.2.1. Elección de los costos de penalización

Inicialmente, se realizó el alineamiento entre los *contigs* y las secuencias de ADN de cada uno de los siete cromosomas, utilizando valores arbitrarios para d y e (*open* y *extended gap*). Con base en el mejor alineamiento, se clasificó cada uno de los *contigs* en cada cromosoma. Se comprobó experimentalmente que al hacer un cambio en alguno de los dos valores (d o e), algunos *contigs* arrojaron alineamientos completamente diferentes. La tabla 4-2 presenta las diferentes clasificaciones obtenidas para cuatro secuencias diferentes de *contigs*. Cada una de las letras representan el cromosoma con el cual se obtuvo el mejor alineamiento con Smith-Waterman. La última fila es la información obtenida al utilizar BLAST. El valor “Sin Ident” significa que para ese *contig* con valor $w=28$; no se encontró coincidencia alguna en la base de datos “nt/nr”. Este es el valor por defecto utilizado por BLAST en la etapa de ensemillado.

La tabla (4-2) demuestra la dependencia de los alineamientos con la selección de valores para d y e . Debido a la variabilidad en los alineamientos obtenidos, se implementó el algoritmo que permite elegir estos valores en base a los alineamientos obtenidos entre las secuencias de los cromosomas y una muestra aleatoria de *contigs* en estudio. Esto motivó el diseño del algoritmo 3, descrito en la sección anterior 3.2.2 (pág. 44). Este algoritmo requiere valores iniciales para d y e , como punto de partida. Se observó que entre 10 y 15 iteraciones el método convergía a un único valor de error, el cual fue calculado utilizando la ecuación 4.1 en cada iteración.

En cada una de las iteraciones del método, el valor inicial del error se reduce hasta llegar a un valor constante y en algunos casos oscila entre dos valores.

<i>Contigs</i> Costos d y e	3269	3545	7189	3832
	Cromosoma con el mejor puntaje			
4 y 0.2	A	B	A	D
4 y 0.4	D	B	G	E
6 y 0.2	E	E	A	D
6 y 0.6	G	C	B	A
8 y 0.2	E	E	A	G
8 y 0.6	E	C	E	A
10 y 0.1	D	A	C	E
10 y 0.5	B	D	E	A
10 y 2	G	A	F	A
10 y 3	E	E	F	A
10 y 5	F	E	F	A
BLAST, w=28	G	Sin Ident	Sin Ident	A

Tabla 4-2: Variabilidad en los alineamientos utilizando diferentes costos de penalizaciones

$$c_error \leftarrow \frac{1}{tam(gaps)} \sum_{g \in gaps} [(d + (g - 1)e) - \log(g)]^2 \quad (4.1)$$

Algunas de las ejecuciones del algoritmo se representan en las dos figuras siguientes. La primera figura 4-2, representa el valor del error en cada iteración utilizando los valores iniciales, $d=2$ y $e=0.5$. En esta se observó que luego de la octava iteración el valor del error oscila entre dos valores.

Por otra parte, la figura 4-3 ilustra el resultado obtenido a partir de los valores iniciales, $d=5$ y $e=0.5$. A diferencia del caso anterior, en esta se observa que el error converge a un único valor después de la séptima iteración. Para el cálculo del error, se realizó el alineamiento con el algoritmo de Smith-Waterman, alineando una muestra aleatoria de 100 *contigs* contra cada uno de los 7 cromosomas, conforme a lo explicado en la sección 3.2.2 (pág. 44).

En cada iteración se almacena el error y los nuevos valores para d y e , los cuales se obtuvieron por regresión lineal entre g y $\log(g)$, donde g es la longitud del *gap*. La

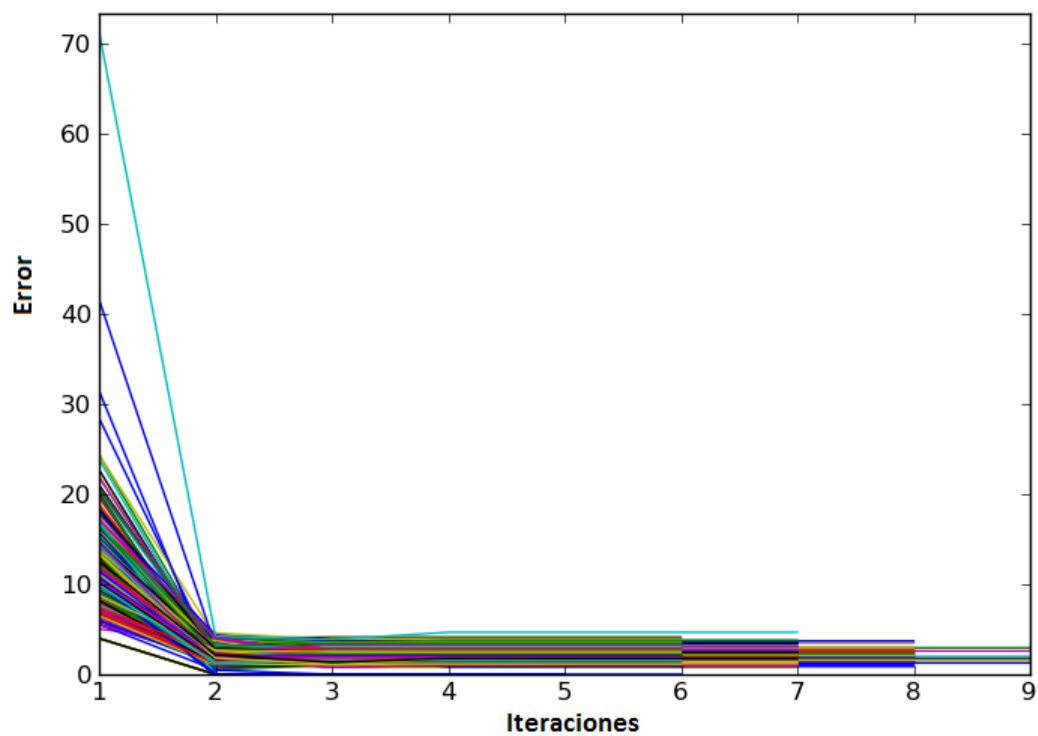


Figura 4-2: Comportamiento del valor del error con valores iniciales $d=2$ y $e=0.5$

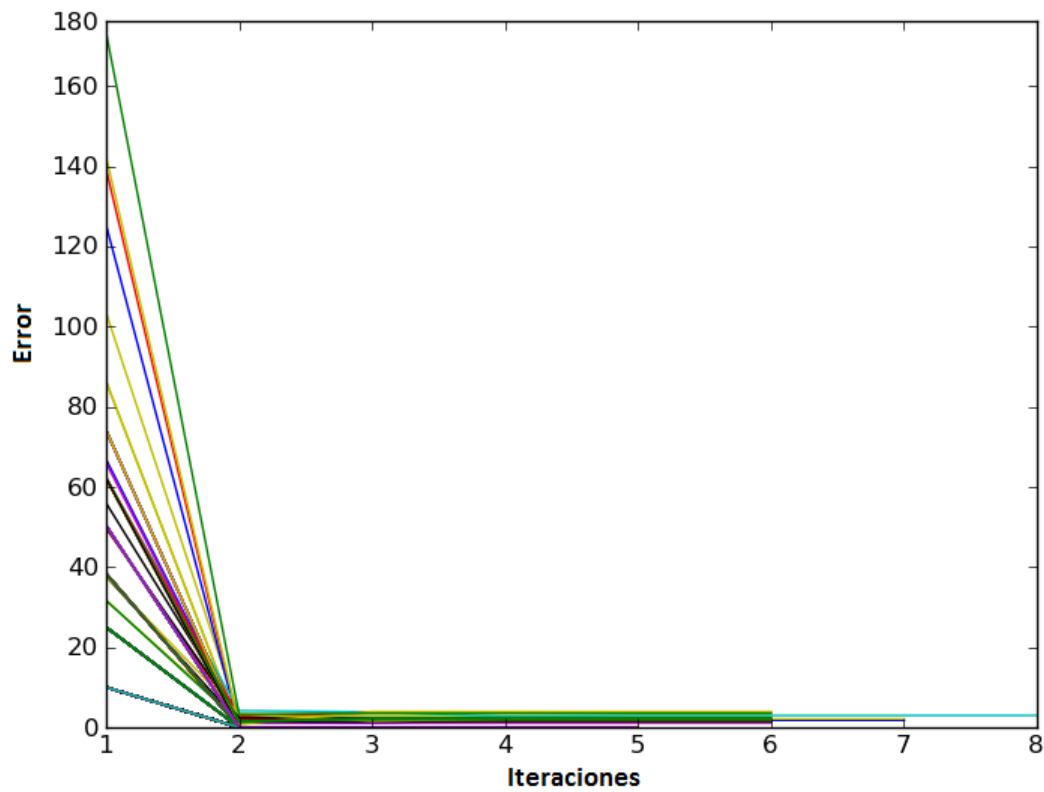


Figura 4-3: Comportamiento del valor del error con valores iniciales $d=5$ y $e=0.5$

pendiente de dicha regresión es el nuevo valor de e y el valor del corte con el eje y es el nuevo valor para d . En el momento que el valor del error comienza a repetirse, el algoritmo se detiene y retorna los últimos valores calculados para d y e . Las gráficas 4-4 y 4-5 representan los valores finales para d y e , encontrados en cada uno de los alineamientos entre los 100 *contigs* y los cromosomas de la especie.

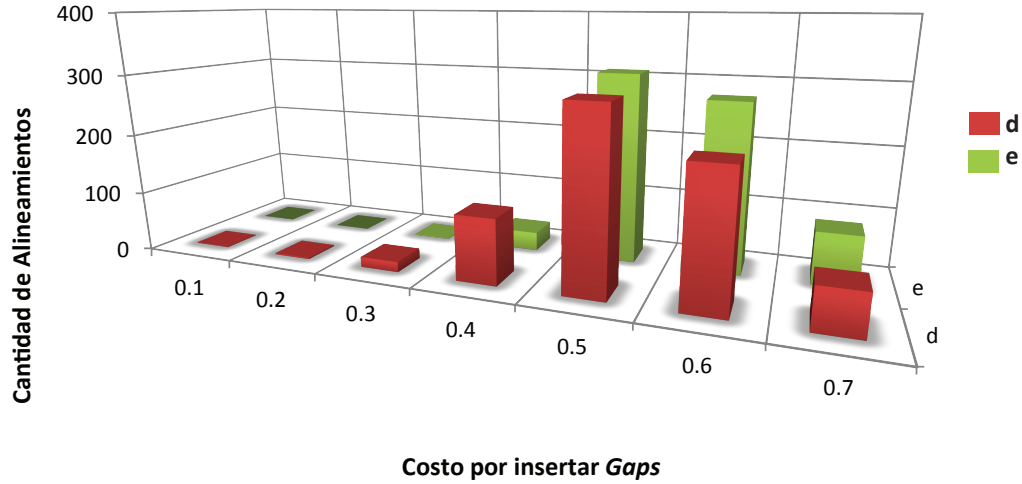


Figura 4-4: Distribución de los costos de penalización, utilizando los valores iniciales $d=0.5$ y $e=0.5$

La primera gráfica muestra los resultados obtenidos por el algoritmo, utilizando los valores iniciales $d=0.5$ y $e=0.5$. En la segunda gráfica se representan los costos de penalización encontrados utilizando los valores iniciales $d=2$ y $e=0.5$. Con base en los resultados presentados anteriormente y el comportamiento estadístico de los mismos, se determinó que los valores apropiados para el *open* y el *extended gap* son $d=0.5$ y $e=0.5$.

4.2.2. Significado biológico

Con el objetivo de determinar si los puntajes de los alineamientos óptimos resultan suficientemente altos como para no ser considerados obra del azar, para cada caso se generaron 200 secuencias aleatorias para las cuales se calcularon los puntajes. Se contruyó un histograma con los 200 puntajes obtenidos y además se

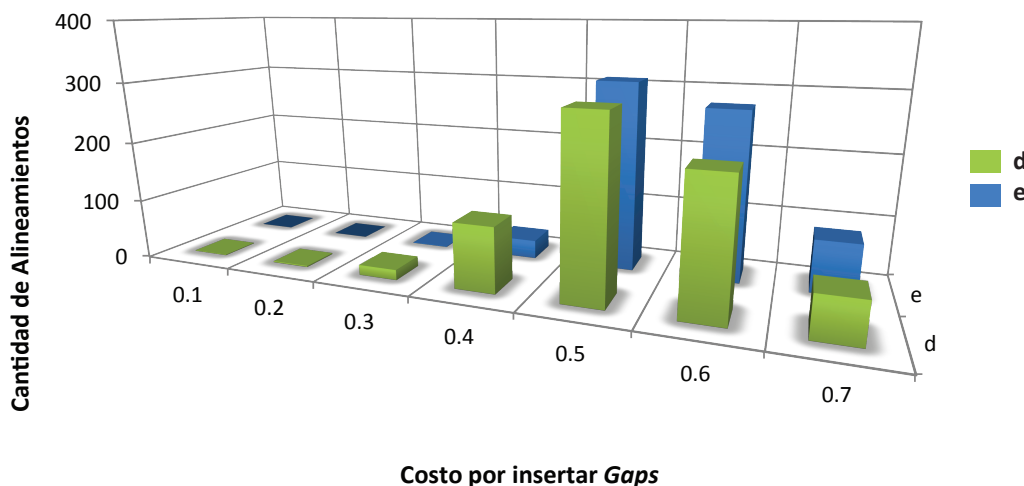


Figura 4-5: Distribución de los costos de penalización, utilizando los valores iniciales $d=2$ y $e=0.5$

aplicó una prueba de bondad de ajuste para determinar si los scores obtenidos de forma aleatoria provienen de una distribución normal. Finalmente se ubicó el *score* obtenido sobre el histograma generado con los *scores* de las secuencias aleatorias, a modo de comparación. Las figuras 4-6 a 4-8 muestran los resultados obtenidos en 3 de los casos.

En los casos presentados, el hecho de que el *score* obtenido para el alineamiento óptimo se encuentre en el extremo derecho de la distribución hace evidente que el *score* inicial óptimo es significativamente mayor a lo esperado para alineamientos al azar.

Luego de analizar los alineamientos realizados entre la muestra de *contigs* y los cromosomas, se encontró que el *score* calculado se encontraba en ó muy cerca del extremo derecho del histograma, siendo este el comportamiento deseado siendo este comportamiento el deseado y ajustándose a lo explicado en [4, 6]. Con base en estos resultados estadísticos se puede afirmar que los alineamientos óptimos obtenidos no fueron por producto del azar. La hipótesis de que estos alineamientos correspondan a la acción de otros factores, como algún tipo de relación biológica se hace más sustentable. Por otra parte, una cantidad inferior del total de los alineamientos

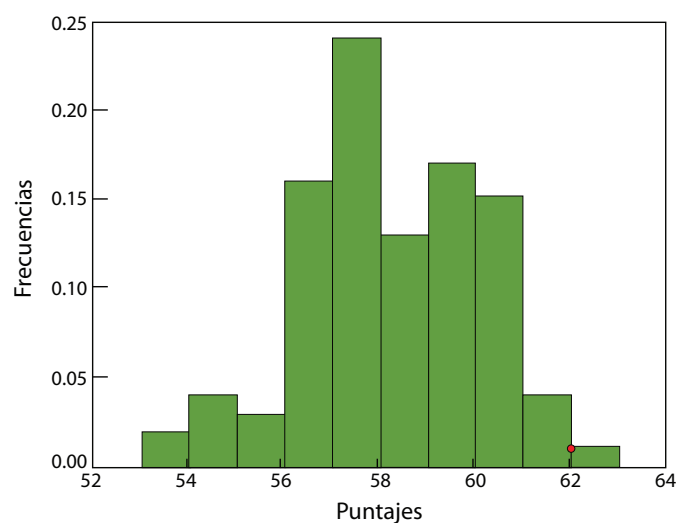


Figura 4-6: Distribución de frecuencias entre el *contig* Node-919577 y el Cromosoma A

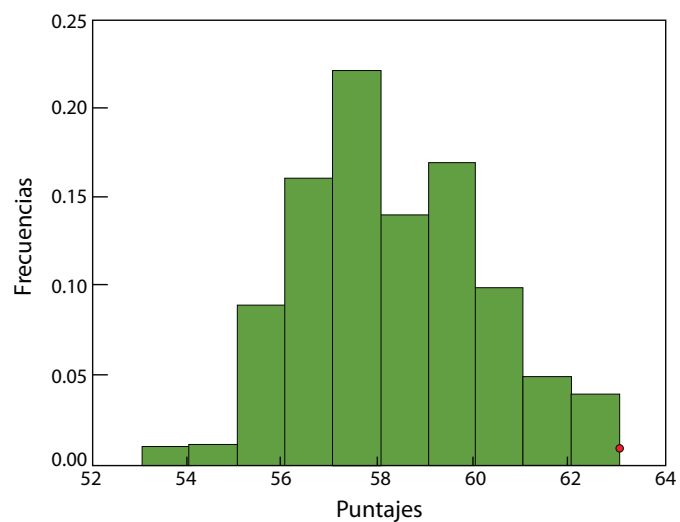


Figura 4-7: Distribución de frecuencias entre el *contig* Node-919577 y el Cromosoma C

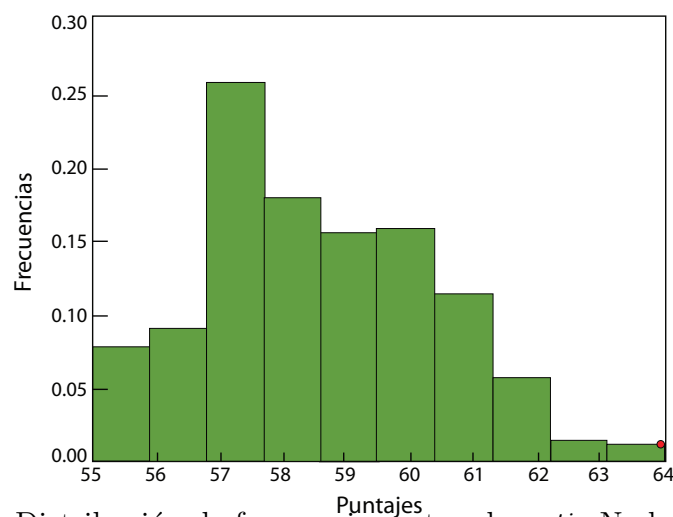


Figura 4-8: Distribución de frecuencias entre el *contig* Node-919577 y el Cromosoma F

realizados sobre la muestra, tuvo un comportamiento similar al observado en la figura 4-9. Allí se observa que el *score* inicial no se encuentra en el extremo derecho de la distribución. En este caso, la hipótesis de que el alineamiento responde a factores biológicos, es menos sustentable.

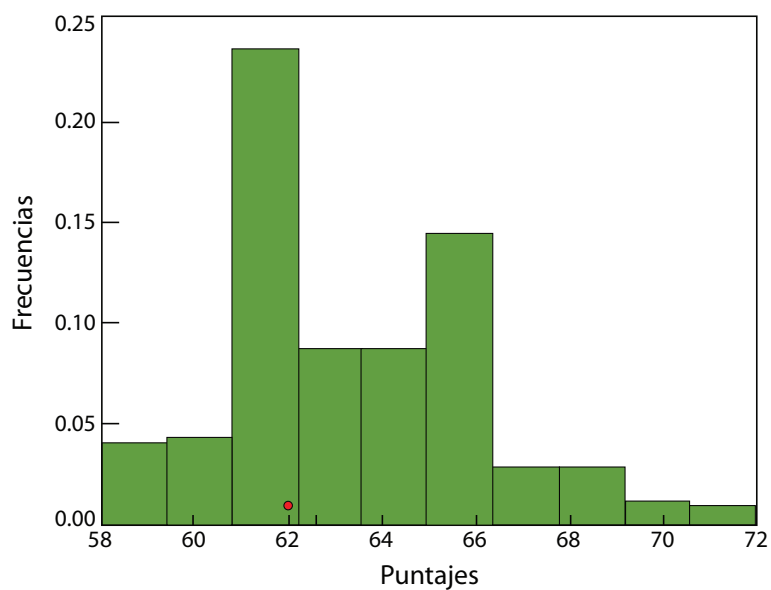


Figura 4-9: Distribución de frecuencias entre el *contig* Node-606252 y el Cromosoma D

Lo porcentajes anteriores, también llevan a concluir que los valores estimados previamente para d y e generan alineamientos con un significado biológico probable en la mayoría de los casos.

4.2.3. Alineamiento de los *contigs*

La clasificación de los *contigs*, se realizó en base al mejor puntaje del alineamiento entre cada *contig* y cada cromosoma. Durante el alineamiento se utilizaron los valores definidos previamente, $d=0.5$ y $e=0.5$. Así como se indicó en la sección 3.2.4 (pág. 48), la clasificación se realizó en tres pasos, los resultados obtenidos se muestran a continuación.

Primera etapa

En esta etapa se realizó el alineamiento utilizando el algoritmo de Smith-Waterman, el cual fue aplicado a los *contigs* relacionados con la primera y segunda cepa de la variante de la especie *D. Hansenii*. Para la primera cepa se logró clasificar el 85 % de los *contigs* en un único cromosoma y la cantidad restante retornó igual puntaje de alineamiento con más de un cromosoma. Con respecto a la segunda cepa, el 84 % de los *contigs* se clasificaron en un solo cromosoma y el 16 % restante en varios cromosomas. El porcentaje de *contigs* clasificados en las dos cepas es muy similar.

Cepa	Cromosomas								Total
	A	B	C	D	E	F	G	másDe1	
Primera	3,569	3,098	3,541	2,834	4,428	5,282	4,799	4,959	32,510
Segunda	3,702	3,084	3,648	2,827	4,710	5,478	4,951	5,265	33,665

Tabla 4–3: Cantidad de *contigs* clasificados en cada cromosoma para dos primeras cepas

La tabla 4–3 muestra en detalle el resultado obtenido tras aplicar Smith-Waterman en los *contigs* de las dos cepas. En la penúltima columna se encuentra la cantidad

de *contigs* que generaron el mejor puntaje con más de un cromosoma. La cantidad de *contigs* están representados en la figura 4–10.

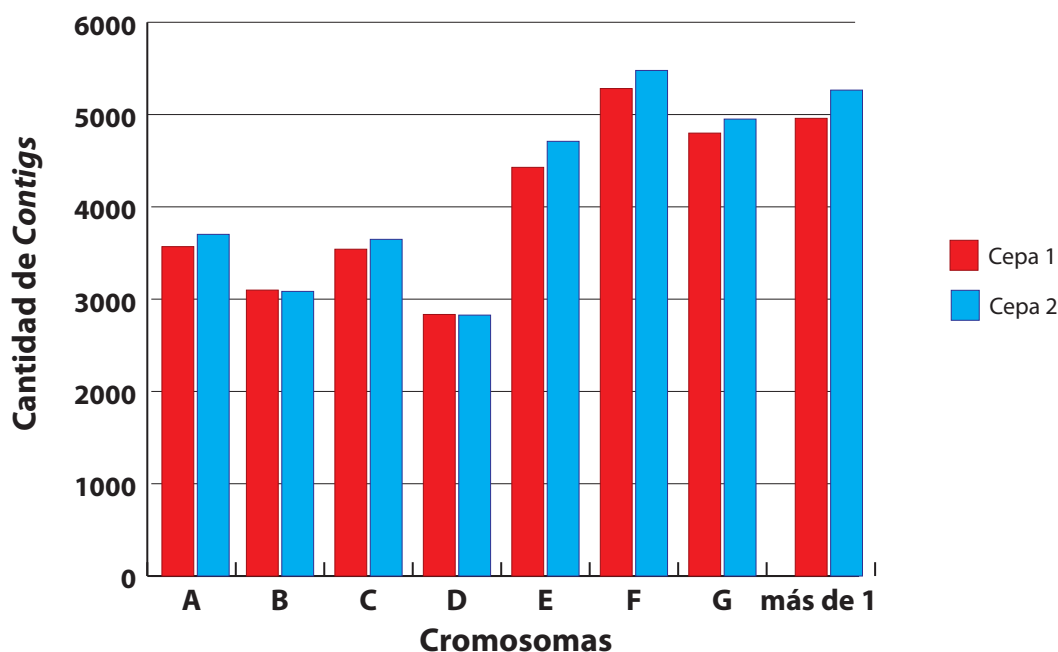


Figura 4–10: Resultados de la clasificación obtenida para las dos cepas luego utilizar Smith-Waterman

Segunda etapa

Para las dos cepas, se hace necesario resolver la ambigüedad de la existencia de *contigs* cuyo mejor alineamiento se obtuvo con más de un cromosoma (másDe1). Para este fin, se repitió el experimento de alineamiento pero esta vez utilizando BLAST. Durante la ejecución de BLAST es posible que no se encuentre coincidencia alguna con ningún cromosoma de la especie *D. Hansenii*. En este caso se hace un segundo intento haciendo el alineamiento con el menor valor para el *word length* ($w=16$). Si después de este cambio persisten *contigs* para los que no hay coincidencia alguna, estos son marcados como “*sin-ident*”, para indicar que no fue posible clasificarlos.

Para la primera cepa, se aplicó BLAST a los 4,959 *contigs* que tenían el mismo puntaje con más de un cromosoma. En la tabla 4–4, se presenta la nueva clasificación

obtenida para esa cantidad de *contigs*. Las dos últimas columnas representan la cantidad de secuencias que siguen aún sin clasificar. Las otras columnas indican la cantidad de *contigs* clasificados entre los cromosomas. Para la primera cepa, ahora la cantidad de secuencias pendientes por clasificar es 410, lo que corresponde a sumar los valores 135 y 275.

Cromosomas									Total <i>contigs</i>
A	B	C	D	E	F	G	másDe1	sin-ident	
427	545	558	605	745	889	780	135	275	4,959

Tabla 4–4: Cantidad de *contigs* clasificados con BLAST para la primera cepa

Con respecto a la segunda cepa, para los 5,265 *contigs* clasificados en más de un cromosoma, se repitió el alineamiento utilizando BLAST. Luego de la ejecución quedaron aún pendientes por clasificar 710 *contigs*. Los resultados relacionados con esta segunda clasificación se encuentran en la tabla 4–5.

Cromosomas									Total <i>contigs</i>
A	B	C	D	E	F	G	másDe1	sin-ident	
477	525	571	565	788	858	771	415	295	5,265

Tabla 4–5: Cantidad de *contigs* clasificados con BLAST para la segunda cepa

En esta parte del proceso, no se tiene aún una clasificación completa de los *contigs* para las dos cepas. Por lo tanto, se hace necesaria una tercera etapa, en la cual se aplica el método de clasificación comparando los valores *e-value* encontrados a partir de las secuencias de los conjuntos *másDe1* y *sin-ident*.

Tercera etapa

En esta última etapa, se clasifican los *contigs* que están aún pendientes (*másDe1* y *sin-ident*). Como se explicó en el párrafo anterior, esta clasificación se logra mediante una comparación entre el *e-value* retornado por BLAST y el calculado a

partir del puntaje obtenido con Smith-Waterman. En la tabla 4–6 se encuentra la clasificación final para los 410 *contigs* restantes de la primera cepa.

Cromosomas							Total <i>contigs</i>
A	B	C	D	E	F	G	
120	90	89	37	45	66	52	499

Tabla 4–6: *Contigs* de la primera cepa clasificados a partir del *e-value*

La cantidad inicial pendiente por clasificar era 410, luego de hacer la clasificación calculando los *e-values* se obtuvieron 499. La cantidad excedida (89), corresponde a aquellos *contigs* en los cuales el mejor alineamiento encontrado con BLAST y Smith-Waterman coincidió en más de un cromosoma durante el proceso de clasificación de los *contigs*.

Por otra parte, en la tabla 4–7 se presenta la clasificación final los 710 *contigs* pendientes por clasificar en la segunda cepa. La cantidad total clasificada a partir de los *e-values* fue 904, la diferencia corresponde con la razón descrita en párrafo anterior.

Cromosomas							Total <i>contigs</i>
A	B	C	D	E	F	G	
178	127	148	98	116	133	104	904

Tabla 4–7: *Contigs* de la segunda cepa clasificados a partir del *e-value*

Luego de clasificar la totalidad de los *contigs* en cada una de las dos cepas, se tienen agrupadas por completo las secuencias en cada uno de los 7 cromosomas. En la tabla 4–8, se presenta la cantidad final de *contigs* clasificados en cada cromosoma para cada una de las cepas.

Tiempos de ejecución

Dado que la complejidad del algoritmo de Smith-Waterman es cuadrática, fue necesario ejecutar este algoritmo en paralelo, aprovechando al máximo la cantidad

Cepa	Cromosomas							Total <i>contigs</i>
	A	B	C	D	E	F	G	
Primera	4,116	3,733	4,188	3,476	5,218	6,237	5,631	32,599
Segunda	4,357	3,736	4,367	3,490	5,614	6,469	5,826	33,859

Tabla 4–8: Clasificación final obtenida para los *contigs* de las dos cepas

de procesadores disponibles en el servidor. Para esto se uso un servidor con 8 procesadores. La figura 4–11 muestra el porcentaje de ocupación de cada procesador del servidor durante la ejecución del algoritmo de Smith-Waterman. En esta se aprecia que los 8 procesadores están al 100 % de utilización, lo que demuestra que la óptima ejecución en paralelo del algoritmo, permitiendo así el uso eficiente de los recursos disponibles.

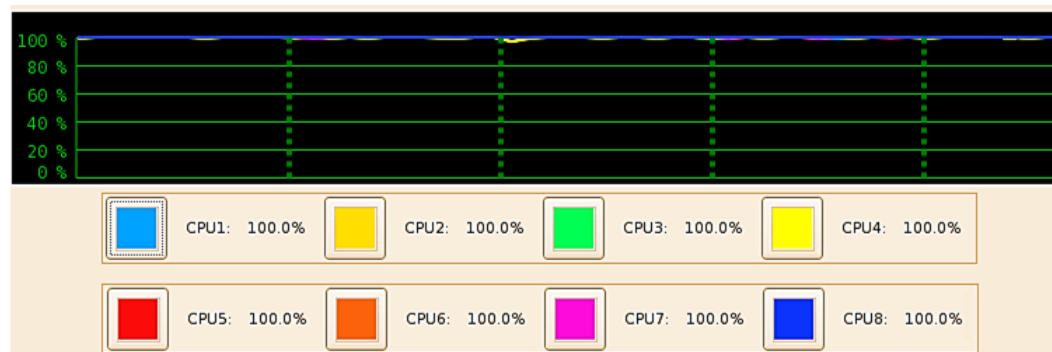


Figura 4–11: Rendimiento del servidor durante la ejecución en paralelo del algoritmo Smith-Waterman

Para realizar el alineamiento de todos los *contigs* en menor tiempo, fue necesario particionar el archivo FASTA en otros de menor tamaño. La partición se realizó en 10 archivos de tamaño similar y sobre cada uno de estos archivos se ejecutó de forma paralela el algoritmo de Smith-Waterman. El alineamiento fue realizado contra cada una de las secuencias de los cromosomas.

Con el ánimo de comparar los resultados obtenidos con BLAST y con Smith-Waterman, se realizó el alineamiento de los 32,510 *contigs* de la primera cepa, utilizando los dos enfoques. El alineamiento con Smith-Waterman fue realizado con el

diseño descrito en el párrafo anterior. En la tabla 4–9 se encuentran los resultados obtenidos por los dos alineamientos, describiendo lo más importante de cada uno.

	BLAST	Smith-Waterman
Procedimiento	Alinear los <i>contigs</i> contra la base de datos “nt/nr”, obteniendo el que arrojó el mayor <i>score</i> y el menor <i>e-value</i>	Alinear cada <i>contig</i> contra cada cromosoma, obteniendo el alineamiento que generó el mayor puntaje, utilizando <i>b</i> y <i>e</i> igual a 0.5
Tiempo de ejecución	En total 18 días alineando la totalidad de los <i>contigs</i> , 32,510 secuencias	Tardó 13 días, ejecutando de forma paralela cada uno de los 10 archivos ya particionados. Cada archivo particionado contenía 3251 <i>contigs</i>
Ventajas	Tiempo inferior de ejecución de la totalidad de los <i>contigs</i>	Clasificación completa de todos los <i>contigs</i>
Desventajas	El 4.8 % de los <i>contigs</i> no arrojó significado con ninguno de los cromosomas	Muy superior en tiempo de ejecución y sensible al valor de los costos de penalización
Observaciones	Se redujó el valor de <i>w</i> de 28 a 16 para obtener resultados en algunos alineamientos	Diferentes <i>contigs</i> retornaron el mismo puntaje para más de un cromosoma

Tabla 4–9: Resultados obtenidos utilizando BLAST y Smith-Waterman sobre los *contigs* de la primera cepa

4.2.4. Reconstrucción de las secuencias

Luego de clasificar los *contigs* corresponde reconstruir cada secuencia sobre la base de su alineamiento al correspondiente cromosoma. Este proceso debe realizarse con la clasificación obtenida para cada una de las dos cepas. Para la primera cepa, se utilizaron los 32,599 *contigs* para reconstruir una sola secuencia. En la tabla 4–10 se muestra la información de las secuencias reconstruidas en cada cromosoma para esta cepa. En estas secuencias la cantidad de letras N insertadas no es superior al 3.3 %. Es importante recordar que la letra N fue insertada en el método del consenso, durante la reconstrucción de la secuencia cuando no fue posible obtener un único nucleótido en las posiciones superpuestas (sección 3.2.5, pág. 51).

Cromosoma	Nucleótidos					Longitud
	A's	C's	T's	G's	N's	
A	493,982	283,939	497,882	285,445	44,089(2,7 %)	1,605,337
B	413,690	239,353	410,307	238,444	42,585(3,2 %)	1,344,379
C	490,255	280,367	486,837	280,278	48,636(3,1 %)	1,586,373
D	391,905	223,169	385,501	225,944	38,915(3,1 %)	1,265,434
E	623,773	347,839	626,886	247,195	60,334(3,2 %)	1,906,027
F	714,110	405,176	706,162	408,328	71,171(3,1 %)	2,304,947
G	629,726	359,526	638,272	355,560	67,059(3,3 %)	2,050,143

Tabla 4–10: Longitud de las secuencias reconstruidas en cada cromosoma para la primera cepa

Con respecto a la segunda cepa, en la tabla 4–11 se presenta la información relacionada con las secuencias reconstruidas. La cantidad de letras *N* en cada una de las secuencias es inferior, en este caso es el 3.4 %, el cual es muy similar al obtenido en la cepa anterior. En los dos casos, este porcentaje representa un mínimo grado de alteración existente entre cada una de las secuencias reconstruidas y la posible secuencia original.

Cromosoma	Nucleótidos					Longitud
	A's	C's	T's	G's	N's	
A	491,281	282,726	496,216	284,301	50,384(3,1 %)	1,604,908
B	412,741	238,680	409,325	237,656	46,048(3,4 %)	1,344,450
C	488,614	279,497	485,696	279,417	52,007(3,3 %)	1,585,231
D	391,315	223,528	384,841	225,878	40,544(3,2 %)	1,266,106
E	621,641	346,853	624,979	346,437	65,833(3,3 %)	2,005,743
F	711,864	403,588	704,522	407,087	77,746(3,4 %)	2,304,807
G	628,674	359,219	636,836	354,814	71,364(3,5 %)	2,050,907

Tabla 4–11: Longitud de las secuencias reconstruidas en cada cromosoma para la segunda cepa

4.2.5. Predicción de genes

El método propuesto en esta investigación está basado en el uso de grafos ORFs (3.2.6, pág. 54). Como alternativa a este método, se utilizó también la herramienta probabilística para predicción HMMgene. A continuación se discuten los resultados

obtenidos con cada uno de los dos métodos.

HMMgene

El programa HMMgene [36], permite predecir genes en secuencias de ADN. La predicción es rápida y está basada en el modelo oculto de Markov (*Hidden Markov Model*), el cual asocia una probabilidad a los resultados obtenidos. El programa utiliza los archivos FASTA de las secuencias reconstruidas de cada cromosoma de la especie *D. Hansenii*.

La salida del programa es una predicción parcial o completa de los genes presentes en la secuencia. Esta información se encuentra en un formato de fácil lectura, donde se especifica la ubicación de todos los genes predichos con la respectiva probabilidad, y la información sobre los exones presentes en la secuencia. Más precisamente, en este archivo se encuentra la siguiente información: el identificador de la secuencia, el nombre del programa, la estructura predicha, comienzo y fin donde se encontró la estructura, probabilidad de la predicción, tipo de secuencia (directa ó complementaria), el marco de lectura y grupo al que pertenece la estructura predicha.

La figura 4–12, es un ejemplo del archivo de salida obtenido, el cual corresponde a un fragmento de los resultados obtenidos para la secuencia reconstruida en el *cromosomaA* de la primera cepa.

La tabla 4–12 muestra parte de los exones y las regiones CDS (*coding sequences*) obtenidas con HMMgene. Esta información está relacionada con la predicción realizada por la herramienta en cada una de las secuencias de los cromosomas de las dos cepas.

La cantidad de regiones CDS, corresponde a aquellas que codifican información genética. En estas regiones pueden estar incluidos exones e intrones. En la tabla anterior también se encuentra la posición donde se encontró el primer exón en cada

```
# SEQ: seq_7_chromosome_A 1605337 (+) A:493982 C:283939 G:285445 T:497882
seq_7_chromosome_A HMMgene1.1e singleex 1492 1746 0.596 + 0 bestparse:cds_1
seq_7_chromosome_A HMMgene1.1e CDS 1492 1746 0.596 + . bestparse:cds_1
seq_7_chromosome_A HMMgene1.1e singleex 8180 9142 0.180 + 0 bestparse:cds_2
seq_7_chromosome_A HMMgene1.1e CDS 8180 9142 0.180 + . bestparse:cds_2
seq_7_chromosome_A HMMgene1.1e singleex 9927 10949 0.123 + 0 bestparse:cds_3
seq_7_chromosome_A HMMgene1.1e CDS 9927 10949 0.123 + . bestparse:cds_3
seq_7_chromosome_A HMMgene1.1e firstex 16948 17266 0.346 + 1 bestparse:cds_4
seq_7_chromosome_A HMMgene1.1e lastex 17548 17858 0.365 + 0 bestparse:cds_4
seq_7_chromosome_A HMMgene1.1e CDS 16948 17858 0.126 + . bestparse:cds_4
seq_7_chromosome_A HMMgene1.1e firstex 27187 27568 0.106 + 1 bestparse:cds_5
seq_7_chromosome_A HMMgene1.1e exon_1 30024 30240 0.321 + 2 bestparse:cds_5
seq_7_chromosome_A HMMgene1.1e lastex 30288 30456 0.773 + 0 bestparse:cds_5
seq_7_chromosome_A HMMgene1.1e firstex 33363 33783 0.303 + 1 bestparse:cds_6
seq_7_chromosome_A HMMgene1.1e exon_1 36760 36852 0.590 + 1 bestparse:cds_6
seq_7_chromosome_A HMMgene1.1e exon_2 36883 37065 0.350 + 1 bestparse:cds_6
seq_7_chromosome_A HMMgene1.1e firstex 44149 44267 0.268 + 2 bestparse:cds_7
seq_7_chromosome_A HMMgene1.1e exon_1 45233 45397 0.044 + 2 bestparse:cds_7
seq_7_chromosome_A HMMgene1.1e exon_2 47047 47148 0.111 + 2 bestparse:cds_7
seq_7_chromosome_A HMMgene1.1e lastex 49911 50718 0.212 + 0 bestparse:cds_7
```

Figura 4–12: Ejemplo del archivo de salida obtenido con HMMgene

Cromosoma	Primera Cepa		Segunda Cepa	
	Cantidad CDS	1er Exón	Cantidad CDS	1er Exón
A	2,747	1,492	2,722	1,383
B	2,362	105	2,257	271
C	2,657	1,927	2,687	1,416
D	2,223	2,049	2,178	3,303
E	3,181	288	3,108	248
F	3,892	829	3,867	674
G	3,401	4,445	3,374	4,216

Tabla 4–12: Información obtenida tras ejecutar HMMgene en las secuencias los cromosomas para las dos primeras cepas

una de las secuencias. Es decir, para el *cromosomaA* de la primera cepa, el comienzo del primer exon esta en la posición 1,492. Según HMMgene, este exón tiene asociada la probabilidad 0.596. De forma similar se encontró la información en los demás cromosomas.

Con el fin de comprobar si el primer exón se encuentra en la posición indicada por HMMgene, se realizó la búsqueda exhaustiva de los exones a lo largo de las secuencias. Luego de realizar esta búsqueda exhaustiva se encontraron exones antes de la posición retornada por HMMgene. Ante esta incertidumbre, se tomó la decisión de realizar una búsqueda exhaustiva en todas las regiones de la secuencia. En la siguiente sección se explica en detalle el método propuesto y los resultados obtenidos luego de realizar la búsqueda exhaustiva.

El método propuesto

El proceso de búsqueda exhaustiva comenzó con la identificación de todas las regiones ORF presentes en la secuencia. Debido a la cantidad de regiones ORFs encontradas, fué necesario limitar la longitud de dichas regiones.

Para esto fué necesario introducir un valor de umbral, el cual permite reducir el tiempo computacional durante el procesamiento de los grafos ORF. Esta mejoría se debe a la notable reducción en la cantidad de regiones ORF al introducir el valor del umbral.

Las tablas 4-13 y 4-14 resumen la cantidad de regiones ORF detectadas en cada cromosoma, en las cuales se observa una reducción notable para diferentes valores de umbral. Entre las diferentes cantidades, representadas en cada columna, se observa una notable reducción entre la cantidad total de regiones ORF (sin umbral) y las otras cantidades cuando se limita el tamaño de la región.

En la tabla 4-13 se presenta la cantidad de regiones ORF superpuestas, contenidas en cada una de las secuencias de cada cromosoma. Por otra parte, en la tabla

Cromosoma	Valor del Umbral					1er Exón
	Sin umbral	3000	2000	1000	500	
A	147,689,694	539,019	357,605	178,450	88,898	313
B	104,678,817	463,782	308,616	153,620	76,454	73
C	139,397,485	534,736	355,981	177,573	88,118	901
D	95,476,662	461,634	305,381	151,589	75,128	1,807
E	239,835,351	693,855	460,387	229,032	114,103	268
F	304,684,361	781,503	519,370	259,270	128,886	121
G	239,457,558	692,760	460,195	228,901	114,073	312

Tabla 4-13: Cantidad de regiones ORF superpuestas, contenidas en cada cromosoma para la primera cepa

4-14 está la cantidad de regiones ORF disjuntas, es decir, la cantidad de regiones ORF que no están superpuestas, lo cual corresponde a cantidades para diferentes umbrales.

Cromosoma	Valor del Umbral				
	Sin umbral	500	1000	2000	3000
A	9,534	9,528	9,504	9,355	9,042
B	8,182	8,182	8,182	8,083	7,822
C	9,641	9,641	9,601	9,446	9,079
D	7,858	7,847	7,839	7,711	7,449
E	11,901	11,897	11,877	11,645	11,271
F	13,688	13,677	13,627	13,343	12,872
G	12,031	12,028	12,014	11,838	11,406

Tabla 4-14: Cantidad de regiones ORF disjuntas contenidas en cada cromosoma de la primera cepa

Algunas de estas regiones pueden compararse con las CDS predichas por HMMgene. Dicha comparación mostró que HMMgene deja de lado un número significativo de las regiones que aparecen en la búsqueda exhaustiva. Adicionalmente, al momento de comparar la posición del primer exón detectado con HMMgene (ver tabla 4-12) y el método aquí propuesto (ver tabla 4-13), se observan resultados diferentes. Para todos los cromosomas, la posición del primer exón encontrado es diferente para los

dos métodos. El método aquí propuesto encontró exones antes de las posiciones indicadas por HMMgene.

Por otra parte, el método aquí propuesto, basado en los grafos ORF, permitió encontrar más regiones codificantes que HMMgene, al igual que la cantidad de exones. Lo anterior hace de este método una alternativa para obtener resultados más precisos, a los obtenidos con el enfoque probabilístico de HMMgene.

El tiempo de ejecución también es otra diferencia entre los dos métodos, pues HMMgene obtuvo los resultados en tan solo minutos, a diferencia del método aquí propuesto, el cual tardó aproximadamente cinco días para procesar más de 1000 regiones ORF en cada cromosoma. Estas regiones corresponden a regiones donde el umbral es igual a 500. Durante la ejecución se observó un crecimiento exponencial con respecto a la cantidad de nodos y *edges* en los grafos en cada una de las iteraciones. Esta es la razón principal por la cual el método aquí propuesto requiere más recursos computacionales y más tiempo de procesamiento al requerido por HMMgene.

En la tabla 4–15 está la cantidad de regiones ORF de longitud inferior a 500, sobre las cuales se realizó la búsqueda de los posibles genes. La cantidad de regiones ORF en cada cromosoma, fue limitada debido al tiempo de ejecución requerido para recorrer todas las regiones presentes en cada secuencia. Para cada una de estas regiones ORF se construyó el grafo, para luego obtener los caminos de mayor peso. Estos caminos representan todas las posibles regiones codificantes.

Cantidad de regiones	Cromosomas						
	A	B	C	D	E	F	G
ORFs	1,339	1,052	1,082	1,020	1,013	1,012	1,006
Codificantes	564	399	421	398	279	410	380

Tabla 4–15: Cantidad de regiones ORF y codificantes obtenidas con el método aquí propuesto para la primera cepa

Posterior a la búsqueda de estas regiones, se realizó una evaluación probabilística de cada región, para seleccionar las más relevantes. Los resultados de esta evaluación se presentan en la siguiente sección.

Evaluación probabilística

Esta evaluación permite filtrar todas las regiones codificantes encontradas con el procedimiento anterior, y escoger aquellas que tienen algún grado de similitud con la secuencias relacionadas con los genes conocidos de *D. Hansenii*.

El método consiste en alinear cada una de las regiones con una base de datos que contiene genes conocidos de *D. Hansenii* utilizando BLAST. De los alineamientos obtenidos se tienen en cuenta únicamente aquellos que tienen un alto *score* al ser comparado con alguno de los genes de *D. Hansenii*. El valor probabilístico se asigna con base en la ecuación (4.2), en la cual *cantAlineamientos* hace referencia al número de alineamientos encontrados con BLAST para la región y *cantGenes* es la cantidad de genes de *D. Hansenii* presentes en dichos alineamientos.

$$probabilidad = \frac{cantGenes}{cantAlineamientos} \quad (4.2)$$

Con el ánimo de aclarar cómo se realizó el proceso de evaluación, en la figura 4–13 [61] se encuentran los alineamientos obtenidos entre una región de ejemplo *seq* y la base de datos biológicas “nt/nr” utilizando BLAST. Para cada alineamiento se tiene la siguiente información: el identificador y descripción de la secuencia, el puntaje del alineamiento (*score*), el *e-value* y un dato adicional sobre la secuencia. En este ejemplo, la letra **G** indica que el alineamiento obtenido hace referencia a un gen y la letra **M** (*Map Viewer*), permite visualizar la información sobre la secuencia del alineamiento.

Sequences producing significant alignments:









Accession	Description	Score	E-value	Links
CR382136.2	Debaryomyces hansenii CBS767 chromosome A complete sequence	903	7.00E-128	
CR382133.2	Debaryomyces hansenii CBS767 chromosome A complete sequence	200	7.00E-48	
XM_456341.1	Debaryomyces hansenii CBS767 DEHA2A00242p (DEHA2A00242g)	200	7.00E-48	
XM_002770675.1	Debaryomyces hansenii CBS767 DEHA2F00308p (DEHA2F00308g)	195	3.00E-46	
CR382138.2	Debaryomyces hansenii CBS767 chromosome F complete sequence	568	3.00E-46	
CR382134.2	Debaryomyces hansenii CBS767 chromosome B complete sequence	390	3.00E-46	
XM_461529.1	Debaryomyces hansenii CBS767 DEHA2F27390p (DEHA2F27390g)	195	3.00E-46	
XM_457683.1	Debaryomyces hansenii CBS767 DEHA2B16742p (DEHA2B16742g)	195	3.00E-46	
XM_457676.1	Debaryomyces hansenii CBS767 DEHA2B16676p (DEHA2B16676g)	195	3.00E-46	
XM_002770328.1	Debaryomyces hansenii CBS767 DEHA2D19404p (DEHA2D19404g)	189	2.00E-44	
CR382135.2	Debaryomyces hansenii CBS767 chromosome C complete sequence	189	2.00E-44	
XM_457688.1	Debaryomyces hansenii CBS767 DEHA2C00110p (DEHA2C00110g)	189	2.00E-44	
XM_458477.1	Debaryomyces hansenii CBS767 DEHA2D00176p (DEHA2D00176g)	113	1.00E-21	

Figura 4–13: Alineamientos obtenidos con BLAST para una región que puede corresponder a un posible gen

A partir de esta información obtenida con BLAST, se calcularon diferentes valores para la región *seq*, los cuales aparecen en la tabla 4–16. Con base en los valores allí calculados y utilizando la ecuación 4.2, la probabilidad de *seq* de ser un gen es de 62 %.

Como se indicó anteriormente este método permite escoger aquellas regiones que podrían ser genes, asignando un grado de pertenencia apoyado en BLAST. La decisión final no puede tomarse sin experimentación bioquímica.

Total de alineamientos encontrados (T)	13
Total de genes en los alineamientos (G)	8
Probabilidad de que <i>seq</i> sea un gen (G/T)	62 %
Mejor <i>score</i> entre <i>seq</i> y uno de los genes	200
Promedio del <i>score</i> para todos los genes	184

Tabla 4–16: Datos encontrados durante la evaluación probabilística de una región ejemplo codificante *seq*

Debido al tiempo de ejecución requerido para trabajar con los grafos ORF, tan solo fue posible procesar algunas regiones codificantes. En la tabla 4–17 se encuentra la información obtenida luego de aplicar la evaluación probabilística a algunas regiones. Estas regiones están contenidas en las secuencias de la primera cepa.

Cromosoma	Cantidad de			Máx. Prob.
	R. ORFs	R. Codificantes	P. Genes	
A	1,339	564	200	62 %
B	1,052	399	147	67 %
C	1,082	421	266	64 %
D	1,020	398	200	67 %
E	1,013	279	170	60 %
F	1,012	410	198	67 %
G	1,006	380	196	60 %

Tabla 4–17: Información sobre las regiones encontradas que codifican información genética en la primera cepa

La segunda columna (R. ORFs), indica la cantidad de regiones ORF procesadas en cada cromosoma. Cada una de estas regiones fue de longitud inferior a 500 caracteres. En la tercera columna, se encuentra la cantidad de regiones codificantes obtenidas luego de crear y analizar el grafo para cada una de las regiones ORF. Los datos de la siguiente columna, son el resultado de aplicar el análisis probabilístico a cada una de las regiones codificantes. La última columna, expresa la máxima probabilidad encontrada para alguna o algunas de las regiones codificantes procesadas. Este valor fue calculado utilizando la ecuación (4.2).

En la tabla 4–18 está la cantidad de regiones encontradas en cada cromosoma, en las cuales la probabilidad fue superior al 50 %. La primera fila indica el intervalo de valores porcentuales y corresponde a la probabilidad de que la secuencia sea un gen. La segunda fila indica la cantidad de regiones que pertenecen a ese intervalo de probabilidad. La información allí contenida está relacionada con los posibles genes encontrados en la primera cepa, a partir de la clasificación de los *contigs*.

	Cromosomas						
	A	B	C	D	E	F	G
Probabilidad (%)	50-62	50-67	50-64	50-67	50-60	50-67	50-60
Cantidad de regiones	158	118	160	159	163	166	94

Tabla 4–18: Porcentajes y cantidad de regiones encontradas de posibles genes presentes en los cromosomas de la primera cepa

De forma similar a lo descrito anteriormente, se procesarán algunas regiones ORF sobre las secuencias relacionadas con la segunda cepa de la especie en estudio. Esta información está representada en la tabla 4–19.

Cromosoma	Cantidad de			Máx. Prob.
	R. ORFs	R. Codificantes	P. Genes	
A	1,044	479	127	64 %
B	1,102	400	158	67 %
C	1,161	490	267	62 %
D	1,373	537	303	67 %
E	1,026	415	229	67 %
F	1,015	496	251	71 %
G	1,018	426	192	60 %

Tabla 4–19: Información sobre las regiones encontradas que codifican información genética en la segunda cepa

Capítulo 5

CONCLUSIONES Y TRABAJOS FUTUROS

5.1. Conclusiones

En este trabajo se desarrolló una propuesta multialgorítmica para el análisis de secuencias biológicas, la cual consiste en aplicar diferentes métodos bioinformáticos sobre secuencias de ADN. Los métodos fueron implementados utilizando el lenguaje de programación *Python*. Durante el análisis de las secuencias, los algoritmos fueron utilizados de forma independiente pero sus resultados fueron combinados con el fin de resolver incertidumbres presentes en los mismos.

Cada uno de los algoritmos desarrollados tiene un propósito específico, dentro de los cuales está la elección de los costos de penalización más apropiados para el alineamiento de secuencias, el cual se realizó en base a relaciones estadísticas sobre una muestra aleatoria de secuencias. Además, utilizando métodos probabilísticos, se validó el significado de dichos alineamientos, con el fin de estimar la probabilidad de aquellos encontrados por razones del azar. Los alineamientos se realizaron con dos algoritmos complementarios: BLAST y Smith-Waterman. La comparación de los resultados obtenidos por ambos métodos fué fundamental para completar la clasificación de los *contigs*.

BLAST arrojó resultados en poco tiempo, gracias al diseño heurístico. Sin embargo, hubo imprecisión en los alineamientos encontrados, pues en un primer intento,

diferentes secuencias no retornaron alineamiento alguno con las presentes en la base de datos “nt/nr”. A diferencia de los resultados obtenidos con Smith-Waterman, el cual fue ejecutado de forma paralela en cada uno de los procesadores del servidor, reduciendo el tiempo computacional, en BLAST el tiempo fue muy superior debido a la programación dinámica, pero se logró alinear todas las secuencias.

Como resultado de la clasificación se obtuvo una sola secuencia en cada cromosoma. Con el objeto de resolver ambigüedades se diseñó, el método de consenso entre nucleótidos, el cual fue de gran utilidad pues después de su aplicación sólo una cantidad inferior al 3.5 % de letras N fueron insertadas en cada secuencia. Esta fue la estrategia implementada para obtener una sola secuencia en cada cromosoma en base a los *contigs* clasificados.

Finalmente, se identificaron los posibles genes presentes en las secuencias utilizando una búsqueda exhaustiva. Durante la búsqueda se utilizaron grafos ORF, implementando algoritmos para analizar este tipo de estructuras de información. La búsqueda de las regiones se realizó con base en reglas definidas y aplicadas a la gramática de cada una de las secuencias. El método permitió encontrar todas las regiones codificantes, pero con la desventaja del crecimiento exponencial en los grafos ORF a medida que aumentaba la cantidad de regiones ORF analizadas. Para limitar el espacio de búsqueda, se introdujo el valor del umbral, el cual permitió reducir la cantidad de regiones ORF analizadas y por ende reducir el crecimiento exponencial del grafo.

Además y a modo de comparación, se utilizó la herramienta HMMgene. En la comparación se observó imprecisión en los resultados de HMMgene, pues el método aquí propuesto encontró más regiones codificantes, debido al análisis exhaustivo a lo largo de toda la secuencia.

Para cada una de las regiones ORF encontradas, se crearon los grafos ORF y posteriormente se encontraron los caminos de mayor costo, los cuales tienen alta probabilidad de ser una región codificante. Estos caminos son las regiones que forman parte del posible conjunto de genes. Luego de predecir estas regiones, se aplicó un método de selección, el cual permitió elegir aquellas con alta probabilidad de corresponder a un gen. Esta evaluación, permitió básicamente asignar un valor porcentual a cada una de las regiones basado en alineamiento con BLAST, entre los genes actuales de *D. Hansenii*. La información de salida de este análisis debe ser interpretada por el experto en biología o genética para validar o rechazar la hipótesis de que esas regiones corresponden a un gen.

Los resultados obtenidos con HMMgene no contenían todas las regiones de interés que se encontraron con el método aquí desarrollado. Debido al tiempo de ejecución requerido para el proceso de los grafos ORF, solamente fue posible procesar algunas de estas regiones para las dos cepas, de la posible variante de la especie *D. Hansenii*. HMMgene en cambio, encontró rápidamente los resultados, siendo esto un factor favorable de la herramienta probabilística.

Durante la implementación y ejecución de la mayoría de los métodos, se observó un comportamiento generalizado. Para obtener resultados de forma rápida en base a la información biológica, es necesario utilizar técnicas heurísticas. Estas básicamente analizan el comportamiento estadístico del contenido de las secuencias, obteniendo resultados con algún grado de probabilidad, lo cual introduce cierto nivel de incertidumbre. Esto a diferencia de los otros métodos, los cuales se basan en programación dinámica y búsqueda exhaustiva y, conllevan a la solución deseada de un problema de optimización. Estos métodos requieren más tiempo de ejecución, pero garantizan la obtención de resultados con mayor precisión.

5.2. Trabajos futuros

A continuación se presentan posibles actividades para ser realizadas más adelante, las cuales están relacionadas con la investigación aquí desarrollada.

Procesar la información relacionada con las cuatro cepas faltantes del espécimen en estudio. Como segunda actividad, permitir la predicción de genes, utilizando diferentes valores de umbral (tamaño del ORF) superior a 500. Para esto, probablemente se requiere del desarrollo de un algoritmo paralelo. Seguido, idear una forma de estimar la función probabilística f en la fórmula, $\gamma(g) = \log(f(g))$ a partir de los datos en estudio con el fin de obtener una estimación más precisa de los costos del *gap*. Por último, realizar la interfaz gráfica que le permita al usuario interactuar con los diferentes algoritmos de forma más natural. Además, incorporar herramientas para visualizar cada una de las estadísticas recompiladas durante el análisis de las secuencias.

APÉNDICES

Apéndice A

MANUAL DEL USUARIO

En esta sección se describen los procedimientos que garantizan el uso correcto de los métodos implementados en el sistema. Asimismo, se mencionan todos los paquetes requeridos que garantizan el funcionamiento del sistema, y la configuración de los archivos de instalación. Finalmente, se explican los archivos de salida obtenidos tras ejecutar cada uno de las métodos implementados.

A.1. Prerrequisitos

Los métodos descritos en la sección 3.2 (pág. 40), fueron implementados en diferentes librerías utilizando el lenguaje de programación *Python*, versión 2.7. A continuación, se mencionan los paquetes y librerías *Python* requeridas para el correcto funcionamiento de los métodos implementados.

A.1.1. Paquetes bioinformáticos

Durante el desarrollo del sistema se utilizaron diferentes paquetes ya existentes, para procesar datos biológicos. Estos paquetes están compuestos por diferentes algoritmos que ofrecen facilidad al trabajar con datos de especies biológicas.

Biopython

Biopython, es un conjunto de herramientas útiles y gratuitas para *Python* y permiten procesar información biológica. Este paquete fue implementado por un equipo internacional de desarrolladores, los cuales siguen actualizando la herramienta. La versión utilizada fue v.1.59, y puede ser descargada de <http://www.biopython.org/wiki/Biopython>.

BLAST

Esta herramienta fue desarrollada por el *NCBI* y es de libre acceso. El mayor uso de BLAST es a través del sitio web de ellos, pero esto es un inconveniente para analizar secuencias de tamaño mayor. Por esta razón, la herramienta puede ser instalada de forma local. Este paquete se encuentra en *C++* y puede ser utilizado desde *Python*. Para el correcto funcionamiento, es necesario descargar la base de datos biológicas contra la cual se desea alinear la secuencia en cuestión.

La versión de BLAST utilizada en esta implementación fue *blast-2.2.23+*, y con respecto a la base de datos, se descargaron todos los archivos relacionados con “nt/nr”. Las instrucciones sobre la configuración y descarga de esta herramienta están disponibles en <ftp://ftp.ncbi.nlm.nih.gov/blast/executables/LATEST/>.

EMBOSS

EMBOSS es un software gratuito desarrollado por el *EMB* (*The European Molecular Biology*). Fue desarrollado con el objetivo de cubrir las diferentes necesidades para el análisis de información de biología molecular. Es posible lograr la interacción desde *Python* con este paquete, lo cual resultó muy provechoso durante el desarrollo del sistema. La versión que se utilizó fue 6.4.0, este se encuentra disponible en <ftp://emboss.open-bio.org/pub/EMBOSS/>.

A.1.2. Librerías de Python

En esta sección se presentan las librerías *Python* para uso específico utilizadas durante el desarrollo. Estas librerías fueron utilizadas, debido a que permiten el uso eficiente de los recursos computacionales en problemas puntuales.

Parallel Python

Parallel Python, es una librería que permite la ejecución en paralelo de código *Python*, funciona de forma similar a *OpenMP* y *MPI*. La versión utilizada fue 1.6.2, es de libre acceso y se encuentra disponible en <http://www.parallelpython.com/>.

NetworkX

Esta librería permite crear y manipular grafos y redes complejas. Además, ofrece diversos algoritmos que permiten trabajar de forma eficiente con grafos. Se utilizó la versión 1.7 y puede ser descargada de <http://www.networkx.lanl.gov.com/>.

NumPy

NumPy, es el paquete fundamental para *Python* que permite el desarrollo de computación científica. Entre la cantidad de utilidades, está la creación de matrices de dimensión N de manera eficiente con capacidad de trabajar con datos genéricos. La versión utilizada fue 1.6.2 y se puede descargar en <http://numpy.scipy.org/>.

SciPy

SciPy, es una librería que permite el uso eficiente de los recursos computacionales para realizar procesos matemáticos, científicos y de ingeniería. La versión sobre la cual se trabajó fue 0.10.1 y esta disponible en <http://www.scipy.org/>.

Matplotlib

Matplotlib, es una librería para *Python* que permite crear gráficos, figuras en 2D, soportando múltiples formatos, entre otras funcionalidades. Esta librería es de libre descarga y esta disponible en <http://www.matplotlib.org/>, la versión utilizada fue 1.1.1.

A.2. Configuración

Todos los métodos propuestos en esta investigación, fueron implementados en diferentes librerías *.py*. Antes de utilizar alguna de estas librerías, es necesario ajustar el archivo de configuración *params.ini*. Este archivo es necesario, pues en este, se encuentra información requerida por algunos de los métodos desarrollados. En la figura A–1, se presenta una imagen de este archivo.

```
[PATH]
BIN_BLAST = /home/students/s106973/ncbi-blast/bin/
[DATABASES]
DB = nt
[SPECIE]
ID = CBS767
NAME = Debaryomyces hansenii CBS767 chromosome
[CHROMOSOMES]
199428773 = Debaryomyces hansenii CBS767 chromosome A complete sequence
199429535 = Debaryomyces hansenii CBS767 chromosome B complete sequence
[LENGTHS]
A = 1249940
B = 1344482
```

Figura A–1: Archivo de configuración *params.ini* utilizando dos cromosomas

En la primera parte del archivo, en el [PATH], se introduce la ruta del *bin* donde se encuentra instalado el paquete BLAST. En la segunda, [DATABASES] hace referencia al nombre de la base de datos biológicas contra la cual BLAST debe hacer los alineamientos. En este caso NT, es el nombre de los archivos relacionados con la base de datos “nt/nr” (*Nucleotide Collection*). En la etiqueta [SPECIE], se introduce

el ID y el nombre NAME de la especie. Estos valores están relacionados con la especie *Debaryomyces Hansenii*. En la siguiente, [CHROMOSOMES], se especifica los datos relacionados con los cromosomas de la especie. Para cada uno se introduce el valor del GI (*GenInfo Identifier*), este es el identificador único de la secuencia. Seguido está la descripción de la especie biológica en estudio. Finalmente, se parametriza la longitud de la secuencia de cada cromosoma, esto en la sección [LENGTHS]

Toda la información relacionada con la especie y contenida en el archivo `params.ini`, fue obtenida del NCBI (<http://www.ncbi.nlm.nih.gov/nucore/?term=CBS767>).

A.3. Librerías del sistema

El sistema computacional desarrollado en esta investigación se compone básicamente de 8 módulos o librerías, las cuales se ajustan a lo descrito anteriormente en la sección 3.2 (pág. 40). Cada una de estas puede trabajar de forma independiente, garantizando la funcionalidad por completo del sistema. En la tabla A-1 se mencionan las librerías, las cuales son explicadas con más detalle a continuación.

Librería	Función
Model_Gap.py	Elección costos de penalización
Multicore_SW.py	Alineamiento con el algoritmo de Smith-Waterman
Multicore_BLAST.py	Alineamiento utilizando BLAST
Test_Scores.py	Validar significado de los alineamientos
Merge_Sequence.py	Clasificar utilizando el método <i>e-value</i>
Build_Sequence	Reconstruir la secuencia a partir de los <i>contigs</i>
Find_Genes.py	Encontrar los posibles genes
Eval_Genes.py	Encontrar probabilidad de que la secuencia sea un gene

Tabla A-1: Librerías implementadas para cada uno de los métodos propuestos

Para la ejecución de las librerías es necesario tener instalados y configurados los paquetes mencionados anteriormente. También son requeridos los archivos `.py` mencionados en la tabla anterior. Como datos de entrada, es necesario tener el archivo multi-FASTA que contiene las secuencias relacionadas con los *contigs*, este es generado por VELVET. Por último, son requeridos los archivos FASTA relacionados con las secuencias de los cromosomas de la especie en estudio.

A.3.1. Model_Gap.py

Esta librería permite obtener los valores adecuados para el *open* y el *extended gap* (sección 3.2.2, pág. 44). Para ello, se requiere la ruta del archivo multi-FASTA que contiene los *contigs*, la ruta donde se encuentran todos los archivos FASTA de los cromosomas y finalmente el valor inicial para el *open* (d) y el *extended gap* (e). La siguiente instrucción permite la correcta ejecución de este módulo:

```
>python Model_Gap.py file_contigs.fasta path_cromosomas d e S Cpus
```

Los valores para d y e pueden ser cualquier valor real. El valor S es el porcentaje de *contigs* sobre los cuales se va a hacer la elección de los valores. Si se desea utilizar todos los *contigs*, el valor de S debe ser 100. El último parámetro, $Cpus$, permite definir la cantidad de procesadores con los cuales se quiere trabajar, con el fin de ejecutar el código *Python* de forma paralela. En caso de que el valor ingresado exceda la cantidad de procesadores disponibles, se utilizan únicamente los disponibles en ese momento.

Archivos de Salida

Luego de ejecutar la librería `Model_Gap.py` los resultados obtenidos se encuentran en la carpeta `OUT_Model_Gap`. En cada uno de los *contigs*, se aplicó lo descrito

anteriormente (sección 3.2.2, pág. 44). En la carpeta están contenidas las gráficas que representan el valor del error encontrado en cada una de las iteraciones. Las figuras obtenidas con esta librería son similares a 4-2.

Además, se encuentran archivos en formato `txt`. Cada archivo contiene la siguiente información: identificador de cada secuencia, el menor valor obtenido para d y e con base en la regresión lineal, el valor de R^2 (coeficiente de determinación de la muestra) de la regresión y por último el menor error encontrado en cada una de las iteraciones. La figura A-2 es una muestra del archivo generado.

Id_seq1;0.43;0.46;0.95;1.68 Id_seq2;0.57;0.59;0.98;1.74 Id_seq3;0.48;0.52;0.96;2.27

Figura A-2: Estructura del archivo generado con `Model_Gap.py`

A.3.2. Multicore_SW.py

Esta librería permite ejecutar en paralelo del algoritmo de Smith-Waterman del paquete *EMBOSS*. Para ejecutar esta librería se requiere: la ruta del archivo con las secuencias en formato multi-FASTA, la ubicación de las secuencias de los cromosomas, los valores para los costos de penalización d y e . Finalmente en `Cpus`, la cantidad de procesadores sobre los cuales se desea hacer la ejecución. La forma de ejecutar esta librería es presentado en la siguiente instrucción:

```
>python Multicore_SW.py file_contigs.fasta path_cromosomas d e Cpus
```

Archivos de Salida

Luego de terminar la ejecución de la librería, en la carpeta `OUT_SW` se encuentran los archivos planos que contienen cada uno de los *contigs* clasificados en cada cromosoma. Tras hacer la ejecución sobre los *contigs* de la primera cepa de la especie, se obtuvieron 8 archivos. Los 7 primeros archivos corresponden a la clasificación lograda en cada cromosoma (`chromosome_x.txt`) y el último archivo

(`several_chromosomes.txt`). Este último archivo contiene aquellas secuencias en las cuales se obtuvo igual puntaje de alineamiento para más de un cromosoma.

Los archivos de salida están en formato multi-FASTA. En cada uno de los identificadores de la secuencia (*header*) se adiciona la posición inicial y final en la cual ocurrió el mejor alineamiento local. Adicionalmente, se encuentra el valor del puntaje del alineamiento obtenido. La figura A–3 es un fragmento de uno de los archivos generados cuando es posible clasificar la secuencia en un único cromosoma.

```
893668_893936_83>idSeq1
CAAGTTTACCCAACGCCGTGGCCGTGATGTAGAATGGAACCGAAGTTTAAAGATGTTGGTA
1389647_1389765_109>idSeq2
TAATATACGATCCTTGAATATCGTTGGAGTCCCCAGTGACCGCTTTATCGCC
1887615_1887842_92>idSeq3
GGGGTTCTGAGTTTCTCTAGTATCATCATAATACATATTGCATATCACTCCTTGAAATATGC
```

Figura A–3: Estructura del archivo generado con `Multicore_SW.py`

Por otra parte, cuando se obtiene el mismo puntaje para en más de un alineamiento, la estructura del archivo cambia. En cada uno de los identificadores de la secuencia, se adicionan todos los cromosomas con los cuales se obtuvo el mismo puntaje. Estos valores están separados por el signo “;”. En la figura A–4 se encuentra un fragmento de este archivo.

```
A_301136_301294_58;B_783978_784124_58>idSeq1
AAATTTGGAAACCGTGATCCTTTTCCACTGTAGCAGTGATGTTCCAACCAAGTTAGTCATGAT
D_5549_5682_70;C_4848_4981_70;F_4399_4532_70;G_20528_20662_70>idSeq1
CTCATCTCCCGGAAGTGTCTGATGACTCCACAGTCGCACCCCATCCAAAGTGGAGAGGTATGGCCGGGAGCG
F_703090_703296_73;D_809141_809325_73>idSeq1
ATTGAATAGTTCATAAACGCGTGATATACAGGAAAGCAATATTTAGCTTTAGGTAT
```

Figura A–4: Estructura del archivo generado con `Multicore_SW.py` cuando existe más de un alineamiento con el mismo puntaje

A.3.3. `Multicore_BLAST.py`

Esta librería permite ejecutar en paralelo la herramienta BLAST. Esta librería requiere: el archivo multi-FASTA con las secuencias, el tipo de archivo a procesar

(1: Si el multi-FASTA fue obtenido con `Multicore_SW.py`, de lo contrario el valor es 0). Cuando el parámetro `Tipo` tiene por valor 1, es cuando se desea procesar el archivo `several_chromosomes.txt`, el cual fue obtenido con la librería anterior. Finalmente, la cantidad de procesadores (`Cpus`) sobre los cuales se desea ejecutar en paralelo la librería.

```
>python Multicore_BLAST.py file_seqs.fasta path_cromosomas Cpus Tipo
```

Previo a la ejecución de esta librería es necesario crear el archivo `contador.txt`. Este archivo debe ser inicializado con valor `-1` antes de cada ejecución. El objetivo de este archivo es llevar un control sobre la cantidad de secuencias procesadas del archivo multi-FASTA. La mayor utilidad, es cuando de forma imprevista, la ejecución es interrumpida, al volver a ejecutar la librería esta comienza a partir de la última secuencia procesada. La base de datos contra la cual se desean buscar los alineamientos fue modificado previamente en el archivo `params.ini`.

Archivos de Salida

Tras ejecutar BLAST, la información obtenida está en la carpeta `OUT_Blast`. Allí están los archivos planos que contienen cada uno de los *contigs* clasificados en cada cromosoma (`chromosome_x.txt`). Además, en los casos que se obtiene el mismo puntaje con más de un cromosoma se encuentran en el archivo (`several.txt`). Finalmente, cuando no se obtiene puntaje con ningún cromosoma, estas secuencias se encuentran en el archivo (`sin_IDENT_BLAST.txt`).

La estructura de los dos primeros archivos es idéntica a los obtenidos con la librería anterior (ver figuras A-3 y A-4). El último archivo, `sin_IDENT_BLAST.txt`, es obtenido cuando el alineamiento no retornó ningún resultado en la base de datos, se encuentra en formato multi-FASTA. Las secuencias contenidas en este archivo

serán las procesadas utilizando el método *e-value*, el cual se explica más adelante y fue explicado en la sección 3.2.4 (pág. 49).

A.3.4. **Test_Scores.py**

Esta librería permite representar gráficamente el significado biológico de una muestra de alineamiento. El primer parámetro requerido es la ruta del archivo que contiene los *contigs* en formato multi-FASTA, seguido de la ruta donde están todos los archivos FASTA de los cromosomas. Los dos parámetros siguientes corresponden a los costos de penalización *d* y *e*. Luego la cantidad de procesadores a utilizar durante la ejecución. El quinto parámetro, *It*, corresponde al número de iteraciones sobre el cual se quiere hacer la distribución estadística de los alineamientos (sección 3.2.3, pág. 46). El último parámetro, *S*, es el porcentaje de *contigs* sobre el cual se quiere aplicar el método. El comando para utilizar esta librería es el siguiente:

```
>python Test_Scores.py f_contigs.fasta path_cromosomas d e Cpus It S
```

Archivos de Salida

Luego de ejecutar esta librería, los resultados se encuentran en la carpeta **Out_Test_Scores**. La figura A-5, muestra la forma en que esta estructurada la información en la carpeta de salida.

Cada uno de los archivos **chromosomes_x_test.txt** se compone de: el identificador de cada secuencia alineada (*x* hace referencia a cada uno de los cromosomas), el puntaje obtenido en el alineamiento y el valor del máximo puntaje encontrado en ese conjunto de alineamientos, estos datos son separados por el signo “;”, así como se ilustra en la figura A-6.

Por otra parte, las imágenes en formato **png**, representan gráficamente la distribución de los puntajes de los alineamiento, al igual que el puntaje encontrado entre

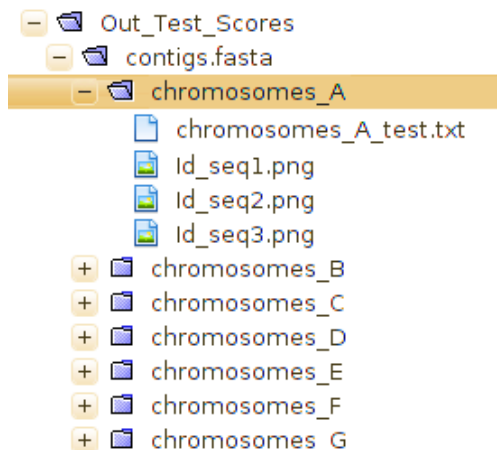


Figura A-5: Datos obtenidos con `Test_Scores.py`

Id_seq1; chromosome_A.txt; 61.0; 64.0
Id_seq2; chromosome_A.txt; 75.0; 74.0
Id_seq3; chromosome_A.txt; 60.0; 62.0

Figura A-6: Estructura del archivo generado con `Test_Scores.py`

cada *contig* y el respectivo cromosoma. Estas imágenes serán analizadas posteriormente por expertos, y así determinar si existe relación biológica entre las secuencias alineadas. La figura A-7 representa la distribución de frecuencias obtenida entre la secuencia `Id_seq1` y el cromosoma A. La información de esa imagen corresponde a la primera línea del archivo utilizado como muestra, presentado en la figura anterior (A-6).

A.3.5. Merge_Several.py

Con esta librería se logra clasificar por completo todos los *contigs*. Hasta este momento existen secuencias en las cuales aun persiste el empate en el puntaje, tras la ejecución del algoritmo de Smith-Waterman y BLAST. Para lograr la clasificación de estas secuencia, se requieren los siguiente archivos: `several_SW.txt`, `several_BLAST.txt` y `sin_IDENT_BLAST.txt`. Estos archivos fueron obtenidos luego de ejecutar los módulos `Multicore_SW.py` y `Multicore_BLAST.py`. La siguiente instrucción permite clasificar las secuencias que se encuentran pendientes:

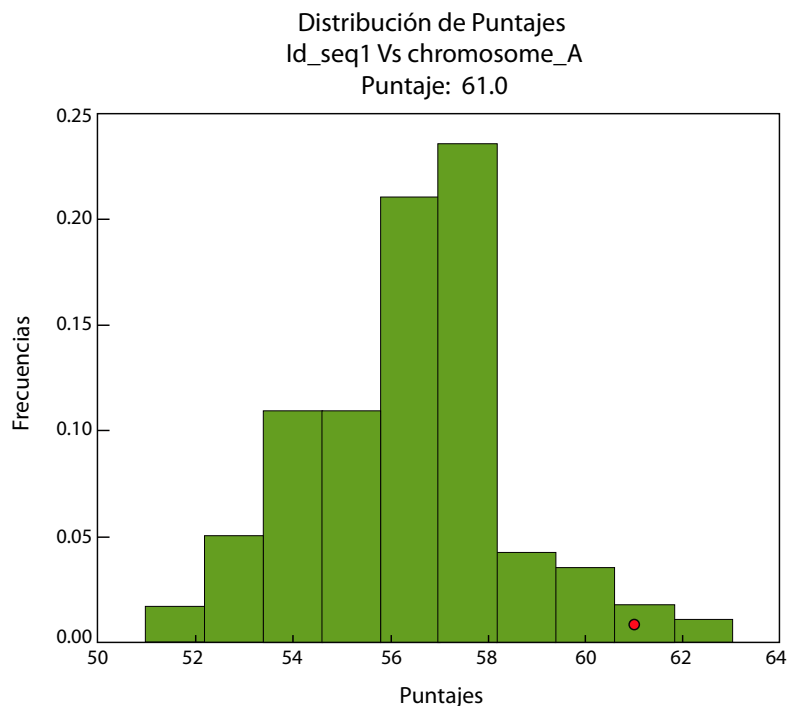


Figura A-7: Ejemplo distribución de frecuencias obtenida

```
>python Merge_Several.py several_BLAST several_SW sin_IDENT_BLAST
```

En esta librería fue implementada el método *e-value*, esta fue la forma establecida para lograr clasificar la totalidad de los *contigs*. Los archivos anteriores deben mantener el formato original con el cual fueron obtenidos, a partir de las librerías `Multicore_BLAST.py` y `Multicore_BLAST.py`

Archivos de Salida

Los resultados obtenidos tras ejecutar este método se encuentran en la carpeta `OUT_JOIN`. Allí están los archivos `chromosome_join_x`, donde *x* hace referencia a cada uno de los cromosomas de la especie. Estos archivos tienen la estructura mencionada anteriormente (ver figura A-3). Con la ejecución de esta librería se logra clasificar por completo todos los *contigs*

A.3.6. Build_Sequence.py

Luego de obtener el archivo multi-FASTA con todos los *contigs* en cada cromosoma, se realiza el proceso de reconstrucción, para formar una sola secuencia. Esta secuencia es reconstruida teniendo como referencia las secuencias de los cromosomas. La forma correcta de utilizar esta librería es la siguiente:

```
>python Build_Sequence file_contigs.fasta file_cromosoma_x.fasta umbral
```

El primer parámetro hace referencia al archivo en el cual se encuentran todos los *contigs* para el cromosoma *x*. El segundo parámetro, es la ruta del archivo en formato FASTA relacionado con la secuencia del cromosoma *x*; estas secuencias son obtenidas del *NCBI*. El último parámetro hace referencia al valor del umbral. Este valor permite determinar cuando el número de ocurrencias de un nucleótido es “muy superior” a las demás cantidades, eso es calculado en el momento del consenso (ver sección 3.2.5, pág. 51). Este valor es porcentual y para realizar el consenso sin tener en cuenta el umbral, es suficiente con colocar 0.

Archivos de Salida

Luego de ejecutar esta librería, en la carpeta `OUT_SEQS` se encuentran los tres archivos: `file_contigs.eNs`, `file_contigs.Gaps` y `file_contigs.Seqs`.

El primer archivo, `.eNs`, contiene los intervalos en los cuales fue necesario colocar la letra *N*, esto es debido al empate obtenido al momento del consenso. En el otro archivo, `.Gaps`, están todos los intervalos donde fue necesario extraer la información de la secuencia del cromosoma. Finalmente, en el archivo `.Seqs`, se encuentra una sola secuencia de ADN, la cual corresponde a la secuencia final obtenida a partir de los *contigs*.

A.3.7. Find_Genes.py

Esta librería permite encontrar las secuencias que pueden formar parte del conjunto de genes de la especie. Para la búsqueda se requieren diferentes parámetros. El primero, es la secuencia que fue obtenida a partir de los *contigs*, seguido el valor F , el cual permite procesar archivos multi-FASTA (0: una sola secuencia y 1: más de una secuencia). Además, es necesario la secuencia en formato FASTA del cromosoma original, con el cual fueron clasificados los *contigs*. Los dos siguientes parámetros d y e , son los valores de los costos de penalización para realizar el alineamiento. El valor de `Cpus` es la cantidad de procesadores a utilizar y el umbral T , determina el tamaño máximo de las regiones ORF a buscar a lo largo de la secuencia. Cuando T vale -1 , se buscan todas las regiones ORF presentes en la secuencia.

```
>python Find_Genes.py f_contigs.Seqs F file_chrom_x.fasta d e Cpus T
```

Para el correcto desarrollo de esta librería, fue necesario implantar los siguientes métodos: `find_exons.py`, `alignments.py`, `algs_graphs.py`. El objetivo principal del primero es buscar todas las regiones ORF y los genes presentes en cada secuencia. En el segundo se tienen implementados los algoritmos de Smith-Waterman y BLAST. Y en el último, están los algoritmos requeridos para el manejo de grafos. Lo anterior es una descripción general de cada uno de los tres métodos que son utilizados desde `Find_Genes.py`.

Archivos de Salida

La información obtenida es almacenada en la carpeta `OUT_Genes`. Allí se crean otros dos directorios: `file_contigs` y `Genes`. En la primera carpeta, hay archivos en formato `.edgelist`, en los cuales se guarda información de cada uno de los grafos ORF generados. La estructura de los archivos `.edgelist` es similar a la figura (3–14). En esta carpeta también se encuentra un archivo `.pairs`, en el cual se reportan todos los intervalos de la secuencia en los cuales se encontró una región ORF. Por otra

parte, en la otra carpeta se encuentra el archivo `.Genes`, en el cual están registradas todas las secuencias que podrían ser genes.

```
450; ATG-17127; GT-17462; AG-17517; TGA-17564; ATGAGTTCATCTGTGGCAGTTTTTAATTGA
363; ATG-17646; TGA-18008; ATGAAGAATTGCAGTTTTTTGTGTGTCGAATTGCAGTTTTTT
141; ATG-17424; TGA-17564; ATGTTTTGAGGAAAGCTATGGAAAGTCTTCAGGCTCAATCC
9; ATG-17556; TGA-17564; ATGATCTGA
```

Figura A-8: Estructura del archivo `.Genes` generado con `Find_Genes.py`

La figura A-8 muestra un fragmento del archivo de salida obtenido. Cada uno de los registros de este archivo corresponde a un posible gene. Las secuencias están ordenadas de mayor a menor costo para el camino. El primer número corresponde al *score*, seguido del camino encontrado y el último valor corresponde a la secuencia obtenida a lo largo de dicho camino. Se reportan todos los caminos, esto le permite al experto analizar diferentes todas secuencias con el respectivo puntaje.

La figura A-8 muestra un fragmento del archivo de salida obtenido. Cada uno de los registros de este archivo corresponde a un posible gene. Las secuencias están ordenadas de mayor a menor costo para el camino. El primer número corresponde al *score*, seguido del camino encontrado y el último valor corresponde a la secuencia obtenida a lo largo de dicho camino. Se reportan todos los caminos, esto le permite al experto analizar diferentes todas secuencias con el respectivo puntaje. En cada uno de los ORF también se generan archivos en formato `.TmpGenes`, los cuales están bajo el formato `.Genes` ya explicado. El objetivo de este archivo es tener información a medida que se van procesando las regiones ORF y al final de la ejecución tener la información consolidada en el archivo `.Genes`.

A.3.8. Eval_Genes.py

Con esta librería se encuentra el valor probabilístico que determina si una secuencia puede ser un gen. Para proceder son necesarios los archivos `.Genes` o `.TmpGenes` en los cuales se encuentra la información de los posibles genes. Estos

archivos fueron generados con la librería anterior. El segundo parámetro es la cantidad de procesadores a utilizar y finalmente el valor que determina el tipo de archivos a procesar (1: `.Fasta` y 0: `.Genes` o `.TmpGenes`). La instrucción para procesar los archivos obtenidos es la siguiente:

```
>python Eval_Genes.py path_files_Genes Cpus fastaFlag
```

Esta instrucción también permite validar cualesquiera secuencias en formato multi-FASTA, con el fin de determinar si son posibles genes de la especie en estudio. La información sobre la especie es obtenida del archivo `params.ini`. En caso de procesar este tipo de archivos en el primer parámetro debe ir el nombre exacto del archivo `.Fasta` a procesar y el valor de `fastaFlag` debe ser 1. Por lo tanto, la instrucción sería:

```
>python Eval_Genes.py path_file.FASTA Cpus fastaFlag
```

Archivos de Salida

Luego de procesar la información, en la carpeta `OUT_EVAL` se encuentra el archivo de salida. Cuando la información a procesar se encuentra en formato FASTA, en dicha carpeta se encuentra el archivo `path_file.EVAL`. En el otro caso, cuando se procesan los archivos (`.Genes` o `.TmpGenes`), se crea un archivo `.EVAL` y el nombre corresponde a la fecha de procesamiento.

El archivo `.EVAL` contiene la siguiente información para cada secuencia: nombre del gene, puntaje del alineamiento, promedio de los puntajes, la probabilidad de que la secuencia corresponda a un gen, la secuencia y la posición donde se encontró la región. El procedimiento relacionado con el cálculo de cada uno de los valores fue explicado en la sección 3.2.6, pág. 57). Los registros en el archivo se encuentran ordenados de forma descendente con base en la probabilidad obtenida para cada secuencia.

Bibliografía

- [1] Maiste P. *Probability and Statistics for Bioinformatics and Genetics*. 2006.
- [2] Needleman D. and Wunsch D. A general method applicable to the search for similarities in amino acid sequence of two proteins. *Molecular Biology*, 48:443–453, 1970.
- [3] Smith F. and Waterman S. Identification of common molecular subsequences. *Molecular Biology*, 147:195–197, 1981.
- [4] Waterman S. *Introduction to Computational Biology, maps, sequences and genomes*. Chapman and Hall, 1998.
- [5] Altschul E., Gish W., Miller W., Myers E., and Lipman D. Basic local alignment search tool. *Molecular Biology*, 215:403–410, 1990.
- [6] Durbin R., Eddy S., Krogh A., and Mitchison G. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. 2002.
- [7] Zerbino D. and Birney E. Velvet: Algorithms for de novo short read assembly using de bruijn graphs. *Genome Research*, 18:821–829, 2008.
- [8] Rogic S., Mackworth A., and Ouellette F. Evaluation of gene-findings programs on mammalian sequences. *Genome Research*, 11:817–832, 2001.
- [9] Heckerman D. and Chickering D. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning Research*, 20:197–243, 1995.
- [10] Salzberg S. Locating protein coding regions in human DNA using a decision tree algorithm. *Computational Biology*, 2(3):473–485, 1995.
- [11] Claverie J. Computational methods for the identification of genes in vertebrate genomic sequences. *Human Molecular Genetics*, 6:1735–1744, 1997.

- [12] Genomic exploration of the hemiascomycete yeasts, June 2012.
- [13] Dujon B., Sherman D., Fischer G., Durrens P., Casaregola S., and Lafontaine I. Genome evolution in yeasts. *Nature*, 430:35–44, 2004.
- [14] Stansfield W. *Genetics*. 2002.
- [15] Jayaram B. Supercomputing facility for bioinformatics and computational biology iit, April 2012.
- [16] National human genome research institute, April 2012.
- [17] The university of utah, genetic science learning center, April 2012.
- [18] Mitchell M. Wood R. and Lindahl T. Human dna repair genes. *Science Direct Mutation Research*, 577:275–283, 2005.
- [19] Goto N. Heuer M. Cock P., Fields C. and Rice P. The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic Acids Research*, 38:1767–1771, December 2010.
- [20] Hackenberg M. Rodríguez N. and Aransay A. *Bioinformatics for High Throughput Sequencing*. 2012.
- [21] Maxam AM. and Gilbert W. A new method for sequencing dna. *Proceedings of the National Academy of Sciences*, 74(2):560–564, February 1977.
- [22] Nicklen S. Sanger F. and Coulson AR. Dna sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467, December 1977.
- [23] Metzker M. Sequencing technologies the next generation. *Nature Reviews Genetics*, 11:31–46, 2010.
- [24] Varma C. Shendure J., Mitra R. and Church G. Advanced sequencing technologies: methods and goals. *Nature reviews Genetics*, 5:335–344, 2004.
- [25] Pettersson B. Uhln M. Ronaghi M., Karamohamed S. and Nyren P. Real-time dna sequencing using detection of pyrophosphate release. *Analytical Biochemistry*, 242:84–89, 1996.

- [26] Lundberg J. Pettersson E. and Ahmadian A. Generations of sequencing technologies. *Science Direct Genomics*, 93:105–111, 2009.
- [27] Mardis E. Next-generation dna sequencing methods. *Review of Genomics and Human Genetics*, 9:387–402, 2008.
- [28] Illumina. De novo assembly using illumina reads. Technical Report 770, Illumina, Inc., October 2009.
- [29] Lesk A. *Introduction to Bioinformatics*. 2002.
- [30] Vingron M. and Waterman S. Sequence alignment and penalty choice. *Molecular Biology*, 235:112, 1995.
- [31] Korf I., Yandell M., and Bedell J. *An Essential Guide to the Basic Local Alignment Search Tool - BLAST*. O’Reilly, 2003.
- [32] Dijkstra E. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- [33] Viterbi A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 1967.
- [34] Ungerboeck G. Trellis-coded modulation with redundant signal sets part i: Introduction. *Communications Magazine, IEEE*, 25(2):5–11, february 1987.
- [35] Sanghamitra B., Ujjwal M., and Debadyuti R. Gene identification: Classical and computational intelligence approaches. *IEEE Transactions on systems, man, and cybernetics*, 38(1), 2008.
- [36] Krogh A. Two methods for improving performance of an hmm and their application for gene finding. *In Proc. of Fifth Int. Conf. on Intelligent Systems for Molecular Biology*, 38(1):179–186, 1997.
- [37] Burge C. and Karlin S. Prediction of complete gene structures in human genomic DNA. *Molecular Biology*, 268:78–94, 1997.

- [38] Solovyev V., Salamov A., and Lawrence C. Prediction of human gene structure using linear discriminant functions and dynamic programming. *Third International Conference on Intelligent Systems for Molecular Biology*, pages 367–375, 1995.
- [39] Gelfand M. and Roytberg M. Prediction of exon-intron structure by dynamic programming approach. *BioSystems*, pages 173–182, 1993.
- [40] Howe K., Chothia T., and Durbin R. Gaze: A generic framework for the integration of gene-prediction data by dynamic programming. *Genome Research*, 12:1418–1427, 2002.
- [41] Aaronson J.S., Eckman B., Blevins R.A., Borkowski J.A., Myerson J., Imran S., and Elliston K.O. Toward the development of a gene index to the human genome: an assessment of the nature of high-throughput EST sequence data. *Genome Research*, 6:829–845, 1996.
- [42] Xu Y., Shah M., Einstein J.R., and Uberbacher E.C. An improved system for exon recognition and gene modeling in human DNA sequences. *Proc. Int. Conf. Intell. Syst. Mol. Biol*, page 376–383, 1994.
- [43] Majoros W.H. *Methods for computational gene prediction*. Cambridge University Press, 2007.
- [44] Pavlovic V., Garg A., and Kasif S. A bayesian framework for combining gene predictions. *Bioinformatics*, 18(1):19–27, 2002.
- [45] Fausett L.V. A survey of neural networks. *Electronic Modeling*, 9(6):1045–1078, 1993.
- [46] Boger Z. Artificial neural networks methods for identification of the most relevant genes from gene expression array data. In *Neural Networks, 2003. Proceedings of the International Joint Conference on Artificial Intelligence*, 2003.
- [47] Ying X., Mural R.J., Einstein J.R., Shah M.B., and Uberbacher E.C. Grail: a multi-agent neural network system for gene identification. *Proceedings of the*

IEEE, 1996.

- [48] Rong S., Jeffrey S.C., Ke W., and Nansheng C. Fast and accurate gene prediction by decision tree classification. *Society for Industrial and Applied Mathematics*, 2010.
- [49] Slazberg S., Delcher A., and Fasman K.H. A decision tree system for finding genes in DNA. *Computational Molecular Cell Biology*, 5(4):667–680, 1998.
- [50] Salzberg S., Pertea M., Delcher A., Gardner M., and Tettelin H. Interpolated markov models for eukaryotic gene finding. *Genomics*, 59:24–31, 1999.
- [51] Saetrom P., Sneve R., Kristiansen K., Snove J., Grnfeld T., Rognes T., , and Seeberg E. Predicting non-coding rna genes in escherichia coli with boosted genetic programming. *Nucleic acids research*, 7(10):3263–3270, 2005.
- [52] Rice P. Longden I. and Bleasby A., 2000. EMBOSS: The European Molecular Biology Open Software Suite.
- [53] Zhang F., Qiao x.Z., and Liu Z.Y. A parallel smith-waterman algorithm based on divide and conquer. *Proceedings Fifth International Conference on Algorithms and Architectures for Parallel Processing*, 2002.
- [54] Hsien L., Meng Y., and Cheng Y. A parallel implementation of the smith-waterman algorithm for massive sequences searching. In *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, pages 1–5, San Francisco, CA, USA, 2004.
- [55] Hirschberg D. A linear space algorithm for computing maximal common subsequences. *ACM Communications*, 18(6):341–343, 1975.
- [56] Tao T., May 2010. Standalone BLAST Setup for Unix.
- [57] Altschul S. Generalized affine gap costs for protein sequence alignment. *Proteins: Structure, Function, and Genetics*, 32:88–96, 1998.
- [58] GU X. and LI W. The size distribution of insertions and deletions in human and rodent pseudogenes suggests the logarithmic gap penalty for sequence

- alignment. *Molecular Evolution*, 40:464–473, 1995.
- [59] Benner S., Cohen M., and Gonnet G. Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *Molecular Evolution*, 229:10651082, 1993.
- [60] Altschul S. and Erickson B. Optimal sequence alignment using affine gap costs. *Bulletin of Mathematical Biology*, 48:603–616, 1986.
- [61] Zhang Z., Schwartz S., Wagner L., and Miller W. A greedy algorithm for aligning dna sequences. *Computational Biology*, 7:203–214, 2000.