Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación

"Alineamiento de Múltiples Secuencias Genéticas usando Cómputo Evolutivo"

TESIS

Que para obtener el grado de:

Maestro en Ciencias
(Computación)

Presenta:

Edgar David Arenas Díaz

Directora de Tesis: Dra. Katya Rodríguez Vázquez

México D.F. 2009

A mi padre y a mi madre

Agradecimientos

En primer lugar agradezco a mis padres, David Arenas Martínez y Crecencia Díaz Toledo, porque sin su ejemplo, apoyo y consejo pocas cosas en mi vida serían posibles.

A la Dra. Katya Rodríguez Vázquez y a la Dra. Helga Ochoterena Booth, agradezco su paciencia, guía y enseñanza que fueron parte imprescindible de este trabajo.

En general, agradezco a todas las personas cercanas a mí (familiares y amigos) por su compañía y complicidad en todos los aspectos de mi vida, inlcuida esta tesis.

Resumen

En este trabajo se utiliza un enfoque de cómputo evolutivo para refinar alineamientos de múltiples secuencias genéticas producidos por alguna otra herramienta de alineamiento, MUSCLE 3.6 (MUltiple Sequence Comparison by Log-Expectation) en este caso de estudio partícular. También se propone un criterio de evaluación de alineamientos, GLOCSA (GLObal Criterion for Sequence Alignment), que toma en cuenta todo el alineamiento a la vez e incorpora conceptos de parsimonia.

El alineamiento de secuencias es un problema fundamental de la bioinformática, en el cuál se busca reconocer las secciones que se han conservado en un conjunto de secuencias para alinearlas y reconocer también las diferencias entre ellas. A partir de esto es muy común que se lleven a cabo análisis filogenéticos para hacer hipótesis de su historia evolutiva. Existen varias técnicas para abordar el problema del alineamiento de múltiples secuencias, entre las cuales destaca el *alineamiento progresivo* (heurística utilizada por MUSCLE).

Debido a la relevancia del problema, un criterio para evaluar la calidad de alineamientos de múltiples secuencias sería de gran utilidad. Pero no existe alguno que tome en cuenta conceptos de parsimonia. Esta es la motivación para proponer **GLOCSA**, un criterio diseñado teniendo la parsimonia como guía principal.

Se adaptó un algoritmo genético para utilizar una representación de los gaps (espacios insertados para alinear las regiones homólogas) en un alineamiento como individuos, los cuales se evolucionan utilizando cinco operadores de mutación que inciden sobre los gaps en las secuencias. Tomando los alineamientos generados por MUSCLE 3.6, se realizaron pruebas usando el algoritmo genético propuesto sobre algunos conjuntos de datos, buscando incrementar el valor de calidad del alineamiento indicado por GLOCSA.

El algoritmo genético siempre mejoró los alineamientos iniciales con respecto a GLOCSA, por lo que este enfoque podría ser utilizado para la refinación de alineamientos utilizando GLOCSA como guía.

Abstract

In this thesis, an evolutionary computing approach is used to refine multiple sequence alignments produced by MUSCLE 3.6 (MUltiple Sequence Comparison by Log-Expectation), which is a widely used sequence alignment tool. A novel evaluation criterion is also proposed, GLOCSA (GLObal Criterion for Sequence Alignment), which rates the whole alignment and considers parsimony concepts.

Sequence alignment is a fundamental problem in bioinformatics, in which it is important to identify conserved regions in a set of sequences to align them, and in consequence, to find the differences between them. From the result of aligning a set of sequences, a common next step is to perform a phylogenetic analysis to construct hypotheses of their evolutionary history. There are various sequence alignment approaches, and one of the most popular is progressive alignment (heuristic which is used by MUSCLE).

Due to the importance of the problem, a criterion to assess the quality of a sequence alignment would be greatly appreciated. But there is not such criterion that takes into account parsimony concepts. This is the motivation to propose **GLOCSA**, a criterion designed based on parsimony principles.

A genetic algorithm was adapted to use a representation of individuals which only encodes the gaps (spaces inserted to align the homologous regions) in an alignment, these individuals where modified (evolved) using five mutation operators, which manipulate the gaps in the sequences. Using the output of a progressive alignment program like MUSCLE 3.6 as an initial starting point, the proposed genetic algorithm was used to seek for improvements in several sets of sequences according to the **GLOCSA** quality measure.

The genetic algorithm always found alignments with improved GLOCSA scores, a fact that suggests that it can be used to find refinements to the output of others approaches, having GLOCSA as its guide.

Publicaciones

Helga Ochoterena, Edgar Arenas, Esteban Ricalde, Cristian Segura, Katya Rodríguez-Vázquez , *GLOCSA: a GLObal Criterion for Sequence Alignment.* In Cladistics, Volume 24, Issue 1 Wiley, 2006.

Edgar David Arenas-Díaz, Helga Ochoterena-Booth, Katya Rodríguez-Vázquez , *Multiple sequence alignment using a GLOCSA guided genetic algorithm.* In Proceedings of the 2008 GECCOACM, New York, 2008.

Índice general

Capítulo 1

Introducción a la Biología Molecular

En junio del año 2000, el Centro Sanger en Cambridge, Inglaterra, anunció la realización de un logro extraordinario que sin lugar a dudas marcó un hito en la ciencia: La entrega del primer borrador del genoma humano, el código de 3,000,000,000 de letras que define y distingue al *Homo sapiens* de las demás especies. Este logro fue la culmnación de un esfuerzo mundial que incluyó a 19 importantes centros de investigación en biotecnología, e incuestinablemente definió una época. La contribución que representó a la cienca fue igualmente monumental, sin embargo la contribución al *conocimiento* fue mucho menos clara. Si definimos conocimiento en este contexto como una colección de respuestas a preguntas en la ciencia de relevancia, tales como "¿Qué genes están involucrados con el sistema inmune humano?", "¿Estamos relacionados más cercanamente con los chimpancés o con los gorilas?", "¿Comó sería posible reducir la velocidad del envejecimiento?", entonces la publicación de una secuencia de ADN no contribuye de manera inmediata. Existe un gran camino que recorrer entre la disponibilidad de una secuencia de ADN y la comprensión de esa información. Este proceso de análisis puede ser llevado a cabo con el uso de la *Bioinformática*. La Bioinformática es un campo interdisciplinario donde se unen la biología, las ciencias de la computación, las matemáticas, la estadística y la teoría de la información para analizar la enorme cantidad de datos biológicos para su interpretación y predicción.

Por ejemplo, un *gen* es una secuencia de ADN (típicamente de 100-5000 nucleótidos de longuitud, representados por símbolos), que codifica la manufactura de una molécula en particular, alguna proteína que lleva a cabo una función específica en las células. Con un genoma recién secuenciado, se podría plantear la siguiente pregunta: "¿Cuántos genes existen en este genoma?". Respecto al genoma humano, la respuesta a esta pregunta puede ser de utilidad para identificar diferencias relativas con otros animales, la forma en que evolucionamos, y tal vez nos proporcione la oportunidad de apreciar que *poco* sabemos de nuestro

propio genoma. Pero la identificación de genes en la secuencia de un genoma nuevo no es un problema trivial. Un enfoque usado comúnmente es desarrollar un modelo computacional predictivo a partir de una base de datos de secuencias de genes ya conocidos y usar este modelo para predecir donde es probale que los genes se encuentren en el nuevo genoma. Actualmente, los enfoques en bioinformática para este problema van desde el modelado estadístico hasta técnicas de *aprendizaje de máquina*[1], como las redes neuronales artificiales. Sin duda, el crecimiento explosivo [5] de la cantidad de datos biológicos requiere de las ideas mas avanzadas y poderosas de aprendizaje de máquina. Y en muchos casos, es posible la aplicación de técnicas de cómputo evolutivo para la solución de problemas que surgen con el análisis de la información biológica.

Sin embargo no se puede esperar obtener resultados trascendentes de solamente usar las técnicas de las ciencias de la computación en los datos biológicos, sin más colaboración que compartir los datos . El conocimiento y la experiencia en cierto campo de la biología puede ser incorporado al análisis computacional de diversas maneras para ayudar a desarrollar un enfoque que dé mejores resultados. El inmenso valor que tiene el conocimiento específico en un campo es difícil de sobrestimar, en particular en un campo que es la intersección entre la biología y las ciencias de la computación principalmente. Una combinación ideal se lograría a partir de la colaboración entre expertos de la biología y de las ciencias de la computación, además de personas con la formación necesaria para enlazar los dos dominos del conocimiento. Es atípico que un solo individuo tenga los conocimientos y experiencia necesarios en ambos dominios. Esta unificación ya ha dado resultados, en el rápido avance en el entendimiento del genoma humano, pero aún existen problemas en los campos donde es necesaria la aplicación de nuevas técnicas para darles una mejor solución, o solucionarlos del todo.

1.1. Biología - Ciencia de la vida

La biología como ciencia intenta proveer un entendimiento de la naturaleza de todos los seres vivientes, en diferetes niveles - desde moléculas a células, individuos, grupos, poblaciones y ecosistemas. La célula sin embargo, es casi universalmente aceptada como la *unidad* primaria de la vida. Todas las formas vivientes están compuestas por una célula o una colección de ellas.

Por el momento, el enfoque principal de la bioinformática está en problemas a nivel de moléculas y células. Los problemas en niveles superiores no son de ninguna manera menos importantes; sin embargo, una gran cantidad de esfuerzo en la investigación científica se está dirigiendo a problemas en niveles bajos de la jerarquía. Como resultado, desde hace algunos años se generan volúmenes impresionantes de información de los niveles molecular y celular (e.g. GenBank [5]). La aplicación de la bioinformática ha seguido esta tendencia.

Cada célula contiene un entorno dinámico compuesto de moléculas, reacciones químicas y una copia del genoma del organismo. A pesar de que el ADN

[1] machine learning

es el mismo en todas las células de un organismo, la expresión del genoma puede ser muy diferente en diversas células, lo que da lugar a una dramática especialización durante el desarrollo de un organismo individual. Solo recientemente se han comenzado a conocer los mecanismos que controlan esta especialización. Las numerosas reacciones químicas en la célula son principalmente el resultado de proteínas. La red de interacciones es alimentada por los nutrientes absorbidos por la célula del ambiente circundante. Incluso, ciertos cambios en el ambiente pueden ocasionar cambios considerables en esta red de interacciones.

Históricamente, la vida ha sido dividida y clasificada en *reinos* o *dominios*, basándose en la similitud de la morfología de las células. Los biólogos comúnmente utilizan los términos *eucariontes* y *procariontes*. Los procariontes, son casi sin excepción unicelulares, y no almacenan su ADN dentro de una membrana especial dentro de la célula, mientras que los eucariontes (unicelulares y multicelulares, e.g., peces, anfibios, reptiles, insectos, aves, mamíferos, hongos, etc.) si lo encapsulan en un compartimento llamado *núcleo* y en otros organelos como la *mitocondria* o el *cloroplasto*. (Figura 1.1) [32].

La comparación de la información de secuencias biológicas de una gran variedad de organismos, usando técnicas de bioinformática ha llevado a la apreciación de que existen tres *dominos* principales de la vida en la tierra: *Eukarya* (que son exlclusivamente eucariontes), *Bacteria* y *Archea* [33]. *Archea* y *Bacteria*, son ambos dominios de procariontes y son morfológicamente similares bajo el microscopio. Sin embargo, organismos de *Archea* son conocidos por vivir en condiciones extremas (e.g. ambientes altamente ácidos o básicos o ventosas hidrotermales en el fondo del mar a grandes temperaturas y presiones), en donde otras formas de vida son incapaces de sobrevivir. Ésto generalmente las separa del resto de los seres vivos. Nuestro conocimiento de este dominio de la vida depende de nuestra habilidad de usar la bioinformática para entender como *Archea* sobrevive en estos ambientes extremos.

1.2. El dogma central de la biología molecular

El ADN (ácido desoxirribonucleico) en cada organismo controla las actividades en sus células, especificando y dirigiendo la síntesis de enzimas y otras proteínas. Para hacer esto, un gen (en el ADN) no construye una proteína directamente (los procesos biológicos son realizados principalmente por la acción de proteínas), sino que construye una cadena de ARN como plantilla, que a su vez codifica la producción de proteínas. Este flujo de información en las células se le ha llamado *el dogma central de la biología molecular*, término introducido por primera vez por Francis Crick en 1958 [12] y después re-enunciado en 1970 [13] [32].

El ADN consiste de dos largas cadenas, cada cadena hecha de unidades llamadas fosfatos, azúcares desoxirribosas y nucleótidos (*adenina* `[A]`, *guanina* `[G]`, *citosina* `[C]` y *timina* `[T]`) enlazadas en serie. Para simplificar la notación, se representa comúnmente a las moléculas de ADN simplemente listando los nucléotidos que las conforman, usando los símbolos `[ A,G,C,T ]`. El ADN en

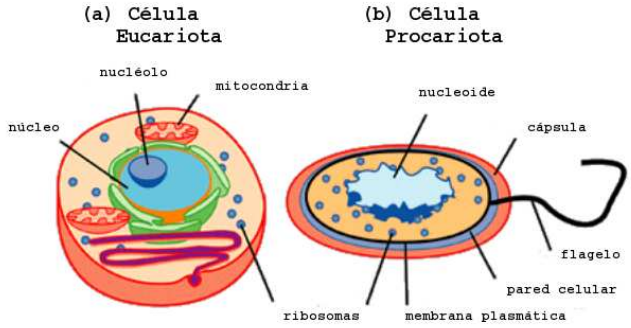


Figura 1.1: Representación de la estructura de una célula eucariota (a) y una procariota (b). En células procariotas el ADN es un componente del *nucleoide*, que es un elemento compuesto de ADN empacado compactamente por ciertas proteínas, pero que no implica membrana nuclear y no se encuentra separado del resto de la célula. En una célula eucariota el ADN se encuentra dentro de un compartimento especial llamado *núcleo* además de en otros organelos como la *mitocondria* o el *cloroplasto*. Adaptado de *Science Primer (National Center for Biotechnology Information, USA)*.

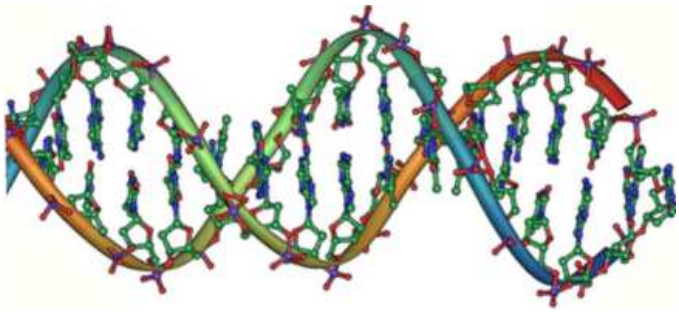


Figura 1.2: Visualización tridimensional computarizada de una molécula de ADN. Adaptado de *Wikimedia Commons*.

cada célula provee las instrucciones completas para construir esa célula (y en el caso de eucariontes multicelulares, todas las células en el organismo). La molécula de ADN es una combinación de dos de estas cadenas desplegándose en una orientación antiparalela para formar una doble hélice siguiendo las reglas de acoplamiento de bases: `[A]` se acopla con `[T]` mientras que `[C]` se acopla con `[G]`. A causa de estas reglas de acoplamiento (también conocidas como reglas de *Chargaff*), las dos moléculas de ADN son complementarias, cada cadena es complementaria estructuralmente a la otra.

Las moléculas de ADN en las células proveen las instrucciones para la producción de ARN y finalmente las proteínas. La transferencia de información desde el ADN hasta una proteína específica (pasando por el ARN) se lleva a cabo de acuerdo al *código genético*. Este código no es universal para todos los organismos (se han observado pequeñas diferencias en ciertos organismos), pero existe un código estandar usado por la mayoría de los organismos, el cual se muestra en la Tabla 1.1. Usando este código, la secuencia de ADN:

```
[ AGTCTCGTTACTTCTTCAAAT ]
```

es primero *transcrita* como una secuencia de ARN usando los nucleótidos *adenina* `[A]`, *guanina* `[G]`, *citosina* `[C]` y *uracilo* `[U]`, que se usa en lugar de *timina* `[T]` en la cadena de ARN:

```
[ AGUCUCGUUACUUCUUCAAAU ]
```

En un organismo eucarionte, esta cadena de ARN típicamente se exporta fuera del núcleo hacia el citoplasma para su *traducción* en la secuencia primaria[2] de una proteína. El ARN es entonces decodificado como una serie de *codones* conformados por tres nucleótidos:

```
[ AGU CUC GUU ACU UCU UCA AAU ]
```

[2] la serie de aminoácidos que la conforman

	UU		AU
UUU	Fenilalanina	AUU	Isoleucina
UUC	Fenilalanina	AUC	Isoleucina
UUA	Leucina	AUA	Isoleucina
UUG	Leucina	AUG	Metionina o Inicio
	UC		AC
UCU	Serina	ACU	Treonina
UCC	Serina	ACC	Treonina
UCA	Serina	ACA	Treonina
UCG	Serina	ACG	Treonina
	UA		AA
UAU	Triosina	AAU	Asparagina
UAU	Triosina	AAU	Asparagina
UAA	Parada Ocre	AAA	Lisina
UAG	Parada Ámbar	AAG	Lisina
	UG		AG
UGU	Cisteína	AGU	Serina
UGC	Cisteína	AGC	Serina
UGA	Parada Ópalo	AGA	Arginina
UGG	Triptófano	AGG	Arginina
	CU		GU
CUU	Leucina	GUU	Valina
CUC	Leucina	GUC	Valina
CUA	Leucina	GUA	Valina
CUG	Leucina	GUG	Valina
	CC		GC
CCU	Prolina	GCU	Alanina
CCC	Prolina	GCC	Alanina
CCA	Prolina	GCA	Alanina
CCG	Prolina	GCG	Alanina
	CA		GA
CAU	Histidina	GAU	Ácido aspártico
CAC	Histidina	GAC	Ácido aspártico
CAA	Glutamina	GAA	Ácido aspártico
CAG	Glutamina	GAG	Ácido aspártico
	CG		GG
CGU	Arginina	GGU	Glicina
CGC	Arginina	GGC	Glicina
CGA	Arginina	GGA	Glicina
CGG	Arginina	GGG	Glicina

Tabla 1.1: El código genético (usado por la mayoría de los organismos). Cada codón (grupo de 3 bases) en el ARN corresponde a un aminoácido. El codón AUG codifica ambos: para la metionina y sirve como sitio de iniciación; el primer AUG en un ARNm es la región que codifíca el sitio donde la traducción de proteínas se inicia.

Aminoácido	**Codificación de una letra**
Alanina	A
Arginina	R
Asparagina	N
Ácido aspártico	D
Cisteína	C
Ácido Glutámico	E
Glutamina	Q
Glicina	G
Histidina	H
Isoleucina	I
Leucina	L
Lisina	K
Metionina	M
Fenilananina	F
Prolina	P
Serina	S
Treonina	T
Triptófano	W
Triosina	Y
Valina	V

Tabla 1.2: Los 20 aminoácidos. Nombres completos y codificación de una letra (como aparecen en las secuencias) se listan.

donde cada codón corresponde a un aminoácido en particular, como se especifica en las Tablas 1.1 y 1.2. Un codón especial llamado *codón de inicio* señala el comienzo del proceso de traducción; este proceso termina cuando uno de los tres codones de terminación o parada es encontrado. En este ejemplo (usando las codificaciones de la Tabla 1.2), se produciría la siguiente secuencia correspondiente a una proteína (asumiendo que es una región interna entre un codón de inicio y uno de terminación)

```
[ SLVTFLN ]
```

Nótese que durante este proceso, se ha transmitido información desde el ADN (la molécula que almacena la información) a ARN (molécula de transferencia de información), y finalmente a una proteína específica (producto funcional y no codificante). Cada uno de estos niveles tiene un complicado sistema de elementos reguladores e interacciones que se han comenzado a comprender apenas desde hace 50 años. El conocimiento de estas interacciones continúa expandiéndose, dando lugar a esperanzas de manipulación de las mismas con diversos fines, la medicina como ejemplo relevante.

Para obtener un mejor entendimiento de este flujo de información, primero se deben secuenciar *genomas* enteros (el contenido completo del material genético) de muchos organismos. Entonces los genes deben ser identificados y nuestro

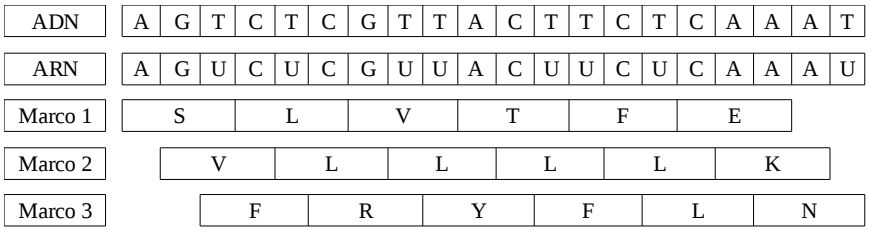


Figura 1.3: Ejemplo de tres posibles marcos de lectura de una secuencia de ADN. Las primeras dos filas muestran la secuencia de ADN y ARN. La traducción del ARN depende del punto de inicio para la identificación de los codones. Los marcos de lectura 1, 2, y 3 comienzan en el primer, segundo y tercer nucleótido respectivamente. Para cada marco la traducción se realiza usando las Tablas 1.1 y 1.2. Adaptado de [19].

conocimiento sobre el código genético usado para determinar el ARN y las proteínas resultantes, para tal vez después inferir una supuesta función de un gen recién descubierto, basándose en la similitud de sus productos (ARN y proteínas) con los de otros genes conocidos. Para realizar este trabajo, se requiere de muchas herramientas computacionales, incluyendo alineamiento de secuencias y de estructuras, reconocimiento de patrones, selección de características[3], métodos para predecir adecuadamente el plegamiento del ARN y proteínas, e incluso la inferencia de propiedades acerca de la sorprendente red de interacciones dentro de una célula.

Como se mencionó previamente, el ADN se puede representar como una serie de símbolos del conjunto $[A, C, G, T]$. En la biología se refiere a cada símbolo en una cadena como una posición de *base* o *nucleótido* (nt) y se refieren a las posiciones complementarias en las dos cadenas de ADN como un *par de bases* (bp). Cuando se intentan descubrir nuevos genes en una secuencia de ADN que no ha sido analizada previamente, se buscan segmentos que se encuentren entre un codón de inicio y otro de terminación y que por lo tanto potencialmente codifican una proteína; segmentos como estos son denominados *candidatos a genes*. Pero muchas veces no se tiene certeza del punto de inicio de la codificación de una proteína y no se conoce con certeza la estructura primaria de ésta. Si el nucleótido de inicio no se tiene definido con exactitud pueden existir tres *marcos de lectura* distintos (como se muestra en la Figura 1.3). Este caso de incertidumbre ocurre de manera concreta en los genomas compactos de bacterias o virus, en donde genes que se traslapan resultan en múltiples marcos de lectura dentro de un mismo segmento de ADN.

Ciertos elementos reguladores (e.g., promotores[4], mejoradores[5]) se encuentran comúnmente flanqueando las secuencias de genes genuinos. Los nucleótidos que conforman estos elementos pueden controlar la expresión de los genes; la

[3] *feature selection*
[4] *promoters*
[5] *enhancers*

presencia de estos elementos también es usada en la investigación para identificar las secuencias que representen genes. En los eucariontes, las regiones codificantes - las regiones que codifican proteínas - representan sólo una pequeña fracción de todo el ADN; el tamaño de esta parte varía considerablente entre diferentes organismos. Incluso dentro de un mismo gen, pueden existir regiones del ADN que llevan a la construcción de algún producto funcional (*exones*) y regiones a partir de las cuales no se deriva ningún producto (*intrones*). Los procariontes contienen mucho menos ADN no codificante en proporción con los eucariontes y tienen su propio sistema de regulación de genes y morfología. Existen muchas diferencias a este nivel entre eucariontes y procariontes, que salen del alcance de esta pequeña introducción, pero es de importacia señalar que los modelos de sistemas en eucariontes no deben ser transportados inmediatamente a procariontes y viceversa.

En muchos eucariontes, las regiones no codificantes están compuestas por un conjunto de múltiples repeticiones contiguas de pequeñas secuencias (*micro satélites*), genes duplicados que han perdido su función codificante (*pseudo-genes*), y segmentos de ADN que son capaces de copiarse y/o insertarse en otras posiciones en el genoma (*transposones*). Por ejemplo, la secuencia *Alu*, un transposón de 280 pares de bases (bp) de longitud, se ha estado replicando y extendiendo su presencia en los genomas humanos por gran parte de nuestra historia evolutiva. Éste y otros transposones similares conforman la categoría llamada *SINE* (abreviatura de *elementos nucleares interdispersos cortos* por sus siglas en ingles[6]). También existen transposones más largos llamados *LINE* (de *elementos nucleares interdispersos largos*[7]). Un ejemplo de LINE es el llamado *L1*, es una secuencia de 7 kilobases[8] (kb), el cual representa casi el 15 % del genoma humano. Los elementos repetitivos claramente representan una gran parte de nuestro genoma. Existe gran interés y un nutrido debate acerca de su función y efectos en nuestra historia evolutiva y nuestros procesos celulares. El restante ADN no codificante o *intergénico* está en su mayoría sin clasificar y estudiar, sin embargo existen razones para no descartarlo completamente y no llamarlo *ADN basura*[9].

1.2.1. Transcripción y Traducción

Como se mencionó previamente, los genes en el ADN son transcritos primeramente como ARN, y después traducidos a proteínas. Los ácidos nucléicos ADN y ARN comparten un lenguaje de símbolos (nucleótidos) muy similar, por lo que se podría decir que el cambio de ADN a ARN es análogo a transcribir un libro escrito en un lenguaje, a otra lengua muy similar. Pero la transferencia de información de ácidos nucléicos a proteínas requiere de una traducción entre dos lenguajes muy diferentes, haciendo uso del código genético.

[6] short interspersed nuclear elements
[7] long interspersed nuclear elements
[8] mil bases o pares de bases
[9] junk DNA

Para transcribir ADN como ARN, un complejo de *pre-iniciación* es ensamblado alrededor de la región promotora que se encuentra en la cadena justo antes del gen que será expresado. Este complejo de proteínas unido al ADN atrae a una proteína muy específica, llamada *enzima ARN polimerasa*, que hace que las dos cadenas del ADN se separen en la región de inicio del gen; entonces la *ARN polimerasa* se enlaza a una de las cadenas. La ARN polimerasa se mueve entonces de izquierda a derecha a través del gen (en biología se refiere a esto como la *dirección 5' a 3'* en referencia a propiedades bioquímicas de los nucleótidos), guiando la producción de una transcripción a ARN. El ARN es construido incorporando el nucleótido complementario correspondiente del conjunto $[A, T, G, U]$ para cada nucleótido encontrado en el gen usando las reglas de formación de parejas mencionadas previamente: A con U; G con C. Este proceso termina cuando la ARN polimerasa encuentra una señal de terminación (un codón de parada) y el producto terminado, el *ARN mensajero* (ARNm) se deja flotando libremente en la célula para después ser exportado o seguir siendo procesado. En células eucariotas, el ARNm se exporta fuera del núcleo hacia el citoplasma de la célula. Ahí encuentra a una estructura de la célula llamada *ribosoma*, que comienza el proceso de decodificación de la información en el ARNm y su traducción al lenguaje de los aminoácidos y proteínas (Figuras 1.4 y 1.5). Una gran variedad de ARN y proteínas participan en este proceso, cuya descripción esta fuera del alcance de esta introducción.

La proteína resultante rápidamente asume su forma nativa tridimensional, (llamada su *estado nativo*) y es entonces cuando está lista para llevar a cabo su función en la célula. El proceso de expresión de genes, descrito a grandes rasgos en los párrafos anteriores, ocurre de manera constante en todas las células vivas, donde, dependiendo del entorno celular se expresan algunos genes y otros no, variando así su nivel de expresión a través del tiempo.

1.2.2. Proteínas

La mayoría de la biomasa de los organismos está compuesta por proteínas, y éstas juegan un papel muy importante en los procesos metabólicos y otros diversos procesos fisiológicos. Cada proteína tiene una estructura tridimensional característica que se encuentra típicamente compuesta por un número de átomos entre 1,000 y 50,000. Las secuencias de aminoácidos de las proteínas son la parte central del flujo de información dentro de las células. Por ejemplo, el 25 % del total de las proteínas del cuerpo humano es *colágeno*. Una molécula de colágeno tiene una función similar a la de un cable fuerte, y grupos de ellas se autoorganizan para dar soporte a estructuras en el cuerpo humano como la piel, órganos internos, huesos, dientes y músculos.

De las decenas de miles de proteínas que existen en las células humanas, sólo muy pocas tienen efectos fácilmente discernibles a nivel del indiviudo. La mayoría de las proteínas determinan características en los humanos de manera sutil. Por ejemplo, un gran conjunto de proteínas llamadas *inmunoglobulinas* son el elemento central del funcionamiento del sistema inmunológico humano; determinan la eficacia con la que se combaten las infecciones de varios tipos. Otro

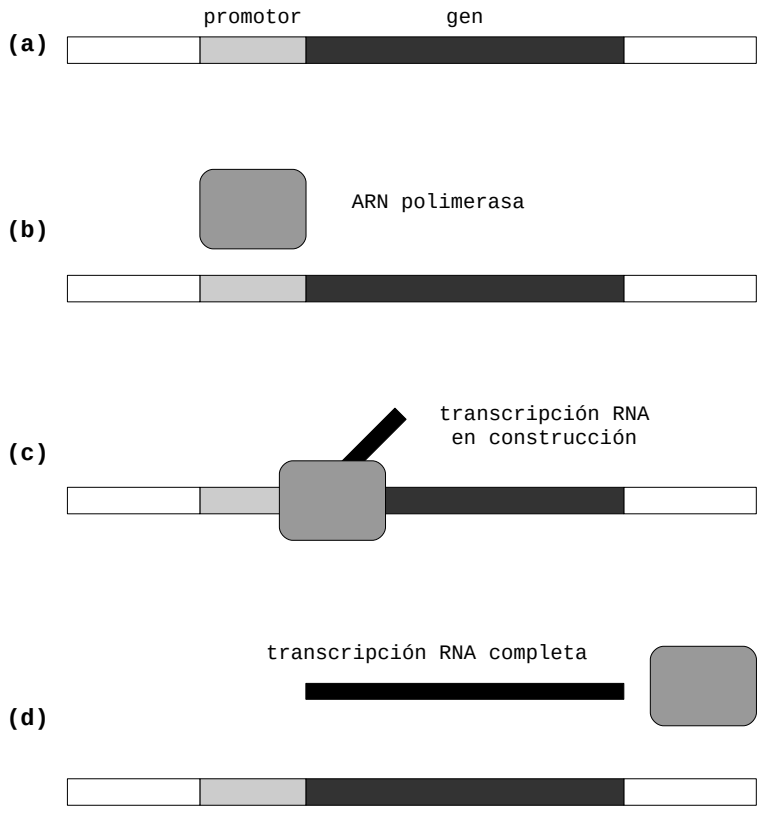


Figura 1.4: Proceso de transcripción. (a) Una sección de ADN que contiene un gen, precedida por una región promotora. (b) Un complejo ARN polimerasa (una colección de proteínas especializadas) es atraído hacia el ADN cerca de la región promotora y comienza a interactuar con el ADN al inicio del gen. (c) La polimerasa ARN se mueve de izquierda a derecha, construyendo gradualmente la transcripción ARN para la secuencia del gen. (d) La transcripción es completada, y ahora se separa el producto de la cadena de ADN y de la polimerasa, quedando flotando libremente. Adaptado de [20].

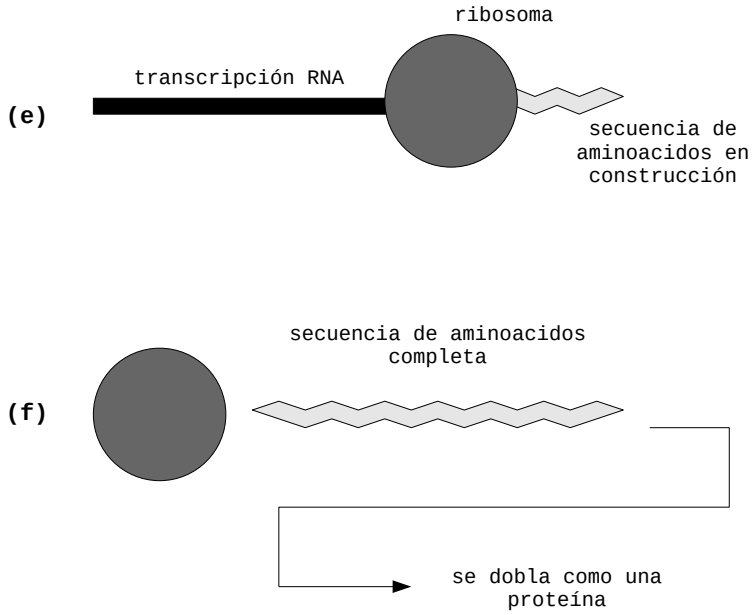


Figura 1.5: Proceso de traducción. (e) La transcripción ARN flota libremente hasta que se encuentra con un ribosoma. El ribosoma utiliza la transcripción para automáticamente producir una proteína. (f) El proceso se completa, y la proteína se dobla en su forma nativa y está lista para realizar su función dentro de la célula. Adaptado de [20].

ejemplo de los sutiles pero decisivos efectos de las proteínas son las llamadas *histonas*, éstas existen en cada célula eucariota y son cruciales para la organización estructural del ADN, y por lo tanto también cruciales para el proceso de interpretación de la información codificada en el ADN. La relación entre las muchas proteínas de una célula conforma una impresionante red de múltiples interacciones que afectan el comportamiento y funcionamiento de cada célula de los organismos vivos.

A pesar de la gran variedad de estructura y funciones de las proteínas, todas comparten un mismo vocabulario estructural. La estructura básica de una proteína se ilustra en la Figura 1.6. Todas las proteínas tienen una "columna vertebral" de átomos de carbono (C) y nitrógeno (N) enlazados secuencialmente formando el patrón $N-C-C$ que se repite. Este patrón o motivo $N-C-C$ es conocido como la *unidad peptídica*, mientras que el enlace que une dos unidades peptídicas (entre un átomo de carbono y otro de nitrógeno) es el *enlace peptídico*; algunas veces las proteínas pequeñas son llamadas también *polipéptidos*. Átomos de oxígeno (O) e hidrógeno (H) se unen al átomo de nitrógeno y al segundo átomo de carbono en cada unidad péptida como se muestra en la Figura 1.6, y en el átomo de carbono central de cada unidad se encuentra un punto de anclaje para cualquiera de los 20 posibles complementos aminoácidos, llamados *cadenas laterales* o *residuos*.

Una proteína, por lo tanto, puede ser expresada por su secuencia de aminoácidos (en este contexto conocido también como cadenas laterales o residuos). Los Biólogos representan cada uno de estos 20 aminoácidos con un símbolo de una letra, como se muestra en la Tabla 1.2. Por ejemplo, la secuencia $PNAYYA$, representa un hipotético segmento de proteína, cuya columna vertebral está compuesta por seis unidades peptídicas; el primer aminoácido es prolina, el segundo asparagina, y así subsecuentemente. De esta manera, cualquier cadena de aminoácidos define totalmente la llamada *estructura primaria* de la molécula de una proteína.

Los detalles estructurales de una proteína están íntimamente ligados a su función en algún organismo, y en casos patológicos, determinan como falla una proteína. La estructura tridimensional de una proteína es crítica para su funcionamiento adecuado y cambios en su forma pueden tener efectos dañinos, neutrales o benéficos para el individuo.

Determinar la forma tridimensional de una proteína (ver Figura 1.7) no es un proceso sencillo [7]. Las cadenas laterales de aminoácidos sobresalen en intervalos regulares, dando soporte y estabilizando la estructura aproximadamente globular. Es sumamente difícil inferir la estructura tridimensional con exactitud apartir de la estructura primaria de una proteína arbitraria. Pero por la inmensa importancia que la forma de las proteínas representa en los procesos biológicos, y la tremenda utilidad que tendría en áreas como el *diseño de drogas*, determinarla es uno de los problemas de más relevancia en la bioinformática en este momento.

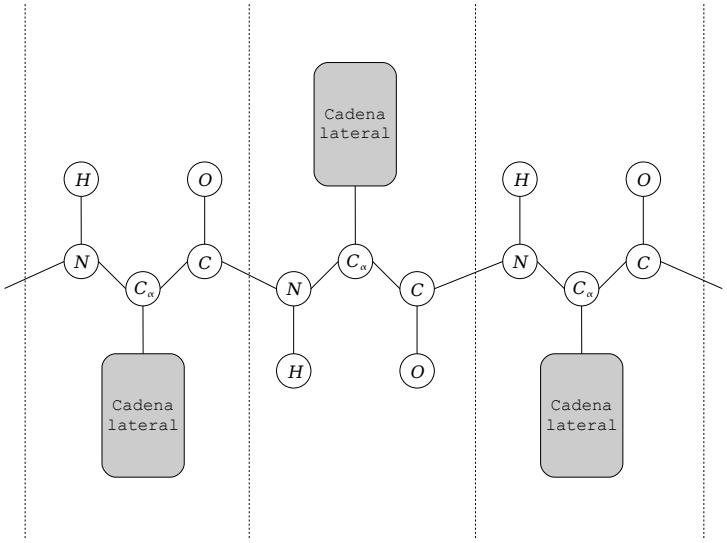


Figura 1.6: Estructura básica de una proteína, los círculos indican átomos de $N = Nitrogeno$, $\{C, C_\alpha\} = Carbono$, $O = Oxígeno$ o $H = Hidrógeno$. Un fragmento de la estructura central de una proteína, está compuesto por repeticiones del patrón $N - C - C$, que alternan su orientación. Una de las 20 *cadenas laterales* se unen al átomo central de carbono (el átomo C_α) del motivo que se repite. Adaptado de [20].

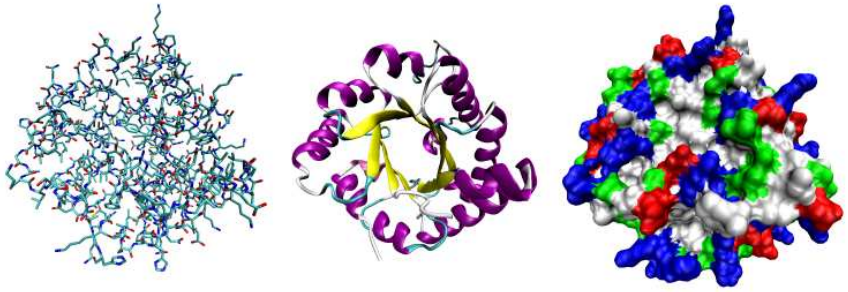


Figura 1.7: Ejemplos de representaciones de estructuras tridimensionales de una proteína. Adaptado de *Wikimedia Commons*, creado originalmente usando datos de PDB [6].

1.3. Redes de genes

Los genes se expresan en diferentes niveles a lo largo de la vida de una célula. Los niveles o grados de expresión de diferentes genes en un mismo genoma puede variar continuamente desde 0 hasta 100,000 proteínas por segundo. Varios factores influyen en el grado de expresión de cada gen, incluyendo el nivel de expresión de otros genes. Resulta útil visualizar el proceso de expresión de genes en determinada célula como una red dinámica, donde los nodos son genes y cuyos enlaces, que tienen pesos definidos con números reales (postivos o negativos), modelan el grado en que la expresión de un gen afecta al nivel de expresión de otro gen. El campo de la *biología de sistemas* busca ofrecer un mejor entendimiento de estas relaciones entre diferentes niveles de expresión de genes [29].

Los enlaces en esta red dinámica tienden a modelar efectos indirectos. Por ejemplo, en un genoma hipotético, un gen identificado como gen 1, codifica una proteína que se une estrechamente a la secuencia de ADN `CTACTG`. La expresión del gen 1 interferiría con la expresión de cualquier otro gen que contenga esa cadena de ADN, además, los promotores que tengan en su cercanía esa cadena también se verían afectados. Sin embargo la expresión de otro gen, identificado como gen 2, puede verse incrementada por el producto del gen 1. La secuencia a la que se une el producto del gen 1 podría localizarse cerca del gen 2, y tendría influencia en esa región del ADN, que podría doblarse de tal manera que alrededor del gen 2 exista un mayor nivel de transcripción y que por lo tanto aumente su expresión. Los niveles de expresión de los genes en las células, pueden ser muy diferentes unos de otros. Cada célula contiene el mismo material genético presente en todas las celúlas de un mismo organismo, pero la historia de la célula y el entorno en el que se encuentra afectan profundamente las redes de expresión de los genes. Los *micro arreglos* (o *chips ADN*) pueden ser usados para medir los niveles de expresión de varios miles de genes en algún momento y entorno precisos. El entorno puede ser cambiado después para hacer que otro patrón de expresión de genes emerja y así poder medirlo. Recientemente esta tecnología se ha hecho disponible y promete ayudar al entendimiento del gran número de interacciones en nuestro genoma.

1.4. Alineamiento de Secuencias

Después de identificar un nuevo gen y las proteínas que produce, típicamente se buscan secuencias similares entre las ya obtenidas previamente. Si son encontradas, estas secuencias similares pueden llevar a indicios sobre la historia evolutiva de los genes y los organismos de donde provienen las secuencias, o dar información sobre la función de una proteína en base a su similitud con otra u otras.

Para buscar secuencias similares existen bases de datos especializadas en secuencias genéticas [5] y de proteínas [6], que se pueden acceder por medio de motores de búsqueda en línea . En estos motores de búsqueda se compara la

secuencia que se da como entrada con las secuencias en la base de datos por medio de *alineamientos*.

Con el objetivo de ilustrar lo que es un alineamiento, se considerará la existencia de dos secuencias de cierto gen en dos organismos diferentes, una recién descubierta y otra ya conocida y almacenada en una base de datos:

```
ATCTCTGGCA
 TACTCGCA
```

Un alineamiento de estas secuencias puede dar el siguiente resultado:

```
ATCTCTGGCA
-T-ACTCGCA
```

En este caso, espacios más comúnmente conocidos como *gaps* ("−") fueron insertados en posiciones donde se asume que alinearán las otras bases con sus homólogas de la cadena superior. Es de notar que en las dos secuencias alineadas no corresponden totalmente todas las posiciones donde no hay gaps. Un alineamiento como éste tiene el objetivo de ofrecer una hipótesis acerca de la relación evolutiva entre las dos secuencias, asumiendo que los organismos de donde se originan provienen de un mismo ancestro común. En este caso los gaps podrían indicar que la secuencia superior se originó por el resultado de *inserciones* en el gen original del ancestro o que la secuencia mas corta es el resultado de *deleciones*[10], o incluso una mezcla de ambas. En otras palabras, el patrón postulado mediante el uso de gaps permite deducir el proceso que le dió origen mediante la insersión o deleción de nucleóticos o aminoácidos. Las posiciones alineadas que no corresponden entre las secuencias simplemente indican mutaciones a partir del ancestro común, en cualquiera de las dos secuencias.

Los alineamientos pueden realizarse entre dos secuencias[11], como el ejemplo anterior, o entre un conjunto de más de dos secuencias[12], aunque la mayoría de las ocasiones los alineamientos de múltiples secuencias se llevan acabo alineando progresivamente pares de secuencias. Incluso pueden ser entre secuencias que representan la estructura primaria de una proteína, es decir la secuencia de aminoácidos que la definen (en este caso se usa la codificación de aminoácidos con una letra mostrada en la Tabla 1.2). En general los alineamientos pueden ofrecer información adicional acerca de importantes posiciones de nucleótidos (y por consiguiente posiblemente aminoácidos) que se han mantenido invariantes en el tiempo durante su evolución, o de posiciones donde han ocurrido cambios durante su historia evolutiva.

Cuando el tamaño de las secuencias y su número aumentan, el número de alineamientos posibles crece de manera importante, lo que hace que se deba buscar un alineamiento lo más similar al mejor ante la imposibilidad de buscar con precisión y eficiencia en todo el conjunto de alineamientos. Esto se hace incluso sin tener una medida convincente, estandar y aceptada sobre la calidad de cierto alineamiento.

[10] como en la biología molecular se llama a la perdida de material genético
[11] *pairwise alignments*
[12] *multiple sequence alignments*

1.5. Bioinformática y Cómputo Evolutivo

El campo de la bioinformática está lleno de interesantes y nuevos retos para las ciencias de la computación. En este momento en el área de la biología molecular existe una gran variedad de proyectos relacionados con los genomas, que están generando importantes volúmenes de información. Pero esta información por si sola es solamente un paso previo para contestar preguntas de inmensa relevancia para las ciencias de la vida. La bioinformática tiene el propósito de proveer los medios para obtener sus respuestas.

Sin embargo, muchos de los problemas a los que se enfrenta la bioinformática son de una magnitud impresionante. El número de soluciones potenciales para un determinado problema (como es el caso del problema de *alineamiento de múltiples secuencias*, al cual este trabajo se enfoca y se introduce en el capítulo 2) puede ser tan grande que hace la búsqueda exhaustiva excesivamente costosa, al grado de resultar impráctica. Ante esta situación es común que se simplifique el problema para que el espacio de búsqueda sea más manejable y poder aplicar métodos exactos. Pero generalmente este enfoque no es el correcto, porque a pesar de que se pueda encontrar la solución óptima para el problema simplificado, esta simplificación en la mayoría de los casos difiere sustancialmente del problema original. Otro enfoque que se podría considerar como más adecuado, es buscar en los vastos conjuntos de soluciones originales pero usando métodos más eficientes, como es la intención de los métodos heurísticos. Uno de estos métodos es el *Cómputo Evolutivo*, que ha sido probado ya en una amplia gama de problemas en ingeniería y sus técnicas han demostrado ser muy eficientes enfrentándose a grandes espacios de búsqueda examinando solo una pequeña fracción de las soluciones posibles. Estas técnicas han sido recientemente aplicadas a problemas de las ciencias de la vida, en el campo de la bioinformática [19]. Específicamente refiriéndose al alineamiento de secuencias y el cómputo evolutivo, se menciona trabajo ya realizado en la sección 2.4. Y es precisamente la aplicación de las técnicas del cómputo evolutivo al alineamiento de secuencias el eje central de esta investigación, donde en primera instancia se realiza la propuesta de una función objetivo para evaluar la calidad de un alineamiento de múltiples secuencias genéticas (la cual se describe en el capítulo 3), para después describir un algoritmo genético que la optimiza (en el capítulo 4) y por ende presenta un método de alineamiento de secuencias alternativo.

Capítulo 2

Alineamiento de Secuencias

Los organismos vivientes son el resultado de su historia evolutiva, lo cual tiene grandes implicaciones para la biología, ya sea a nivel molecular o del organismo entero. En la biología clásica, las estructuras compartidas con un ancestro común son llamadas *homólogas*. Por ejemplo, la mano humana es homóloga a la mano de un chimpancé, la garra de un gato y el ala de un murciélago. Estas estructuras tienen características anatómicas comunes, a pesar de tener funciones muy diferentes entre sí. En contraste, las alas de un murciélago y las de una mosca no son homólogas y son anatómicamente muy diferentes, aún cuando ambas son utilizadas para volar. Este tipo de estructuras son llamadas *análogas*. Homología tiende a implicar similitud de la función apartir de un origen común, mientras que similitud en la función se puede deber a analogía u homología.

En la biología molecular la búsqueda de características homólogas, que en este caso serían residuos en el ADN o secuencias de proteínas, reaparece como el problema de alineamiento de secuencias. Dos secuencias se alinean al escribir los residuos homólogos en una misma posición, uno encima del otro. Lo cual nos puede llevar a un resultado satisfactorio o no, i.e. el alineamiento resultante puede ser convincente y con sustento biológico, o no. Si dos proteínas se alinean satisfactoriamente, es porque tienden a ser similares. Más aún, tienden a tener funciones similares.

Los alineamientos pueden ser de pares o de múltiples secuencias. Los alineamientos por pares pueden ser considerados como un caso especial de los alineamientos múltiples. En la práctica, sin embargo, la complejidad computacional de alinear múltiples secuencias es tan grande que los algoritmos correspondientes no son extensiones directas de los algoritmos para alinear pares de secuencias, ya que estos generalmente involucran técnicas de *programación dinámica* que si bien aseguran encontrar el alineamiento óptimo según determinado criterio, sus propiedades de escalamiento con respecto al número de secuencias a alinear no son las mejores. Es por esto que se han incorporado métodos heurísticos para resolver el problema del alineamiento de secuencias.

2.1. Alineamientos y su Interpretación Biológica

Las secuencias de ADN y las proteínas que codifican, cambian a través del tiempo, evolucionan por medio de la mutación. Los dos tipos más simples de mutación son las mutaciones puntuales, y las inserciones/deleciones, también conocidas como *indels*. Cuando se alinean dos secuencias se ponen los residuos (codificaciones de bases) uno sobre otro y se insertan "−" buscando alinear los residuos homólogos. Si los dos residuos alineados son indénticos se supone que no existió cambio alguno; si son alineados dos residuos diferentes, esta diferencia se puede interpretar como una mutación puntual, en el caso que sea alineado un residuo con un espacio "−", un evento indel es supuesto. Dos "−" alineados juntos no son permitidos, ya que esto representaría que ambas secuencias heredan un indel en esa posición, lo cual si bien es posible, no es determinable sólo con la información que representan dos secuencias. Para los alineamientos múltiples se hacen las mismas suposiciones, ya que los alineamientos de dos secuencias pueden ser considerados un caso particular de los otros.

De esta manera, el alineamiento de secuencias tiene la interpretación biológica de reconstruir la historia evolutiva de las secuencias, insertando "−" para acomodar las posiciones de las secuencias de manera que reflejen las mutaciones puntuales y los indels. Es aquí donde está la complejidad del problema, ya que se pueden postular infinidad de configuraciones que impliquen diferentes conjuntos de mutuaciones e indels, haciendo el espacio de búsqueda de un tamaño considerable, que crece con el tamaño de las secuencias y con el número de éstas que se desea alinear. Es precisamente por esta razón que es común que se realicen los alineamientos de múltiples secuencias de manera progresiva, es decir alineando pares de secuencias progresivamente, para evitar lidiar con todo el alineamiento a su vez, aunque considerar todas las secuencias juntas proporciona más información para alinearlas.

2.2. Alineamientos por Pares

Existen varios enfoques para alinear un par de secuencias, pero las tres principiales estrategias para hacerlo son llamadas *global*, *local* y *traslape* (Figura 2.1).

Un alineamiento global es lo ideal cuando las dos secuencias son homólogas en toda su extensión, ya que se busca alinear las dos secuencias completas. Este tipo de alineamientos es común cuando se desean comparar genes que ya se encuentran identificados y entre los cuales se supone una relación evolutiva, con el objeto de identificar las regiones conservadas y las diferencias entre las secuencias que definen a los genes.

Los alineamientos locales están diseñados para secuencias que tienen regiones limitadas homólogas en su interior, fueron concebidos por la necesidad de alinear secuencias donde existen regiones muy diferentes entre ellas, y en las cuales sólo existen regiones de menor tamaño que se conservan lo suficiente como para realizar un alineamiento confiable únicamente sobre esas regiones.

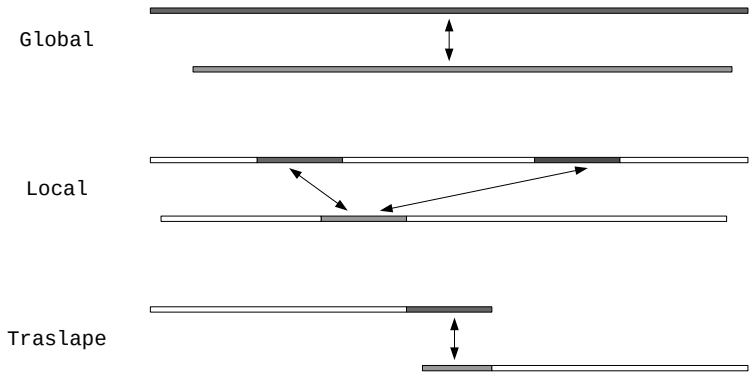


Figura 2.1: Estrategias de alineamiento de dos secuencias.

Año	Autores	Algoritmo	Tipo de Algoritmo
1970	Needleman y Wunsch	Global	Óptimo
1981	Smith y Waterman	Local	Óptimo
1988	Pearson y Lipman	FASTA	Heurístico
1990	Altschul *et al*	BLAST	Heurístico
1997	Altschul *et al*	Gapped BLAST	Heurístico

Tabla 2.1: Hitos en el desarrollo de algoritmos de alineamiento. Adaptada de [23].

Finalmente la estrategia de traslape es usada cuando se sabe que dos cadenas pueden compartir una region homóloga al inicio de una y al final de la otra, técnica usada para secuenciar genomas completos por medio de *secuenciación escopeta*[1], que consiste en secuenciar múltiples fragmentos del genoma para después ensamblarlos en base a los traslapes entre ellos.

Ya sea local, global o por traslape, los alineamientos pueden ser generados por dos tipos de algoritmos: *óptimos* y *heurísticos*. Cronológicamente, como se puede observar en la Tabla 2.1, los algoritmos óptimos fueron desarrollados primero y están basados en una técnica de las ciencias de la computación llamada *programación dinámica*. De esta manera se garantiza que se encontrará el alineamiento mejor con base en el sistema de evaluación y sus parámetros establecidos. Los algoritmos heurísticos, en cambio, se pueden considerar generalmente como procedimientos más rápidos para aproximar a sus contrapartes óptimas. El primer algoritmo de alineamiento óptimo fue publicado en 1970 [34] mientras que FASTA, el primer algoritmo de alineamiento heurístico usado ampliamente [38] fue publicado casi dos décadas después.

[1] shotgun sequencing

A	A	A	A	A	A	-	-	-	-	-	-
-	-	-	-	-	-	C	C	C	C	C	C

Tabla 2.2: Alineamiento trivial

2.2.1. Evaluación de Alineamientos

Para identificar el *mejor* alineamiento, se debe primero tener en cuenta que en términos biológicos se busca alinear residuos homólogos. Además, se supone que la evolución es parsimoniosa, lo que nos lleva a buscar el alineamiento que implique el menor número de eventos evolutivos, es decir mutaciones puntuales e indels. Pero minimizar el número de eventos solamente lleva a encontrar alineamientos triviales, como por ejemplo, insertar un gap inicial para recorrer una secuencia de tal manera que solo queden alineadas bases con "−" (Tabla 2.2).

Debido a esto, en lugar de buscar minimizar el número de eventos evolutivos implicados, se define una función que evalue la similitud de las secuencias alineadas y se maximiza esa similitud.

La unidad mínima en un alineamiento de un par de secuencias es una posición alineada en cada una de las secuencias, es decir, las bases o base y gap que se encuentran en la misma posición. Los modelos clásicos para evaluar un alineamiento se basan en la idea de evaluar cada posición. Al realizar esta evaluación se hace una diferencia importante entre evaluar un par de nucleótidos o aminoácidos o posiciones que forman parte de un indel.

En el caso de los indels, el modelo para evaluarlos más extendido es el de *affine gap model*[2], en el cual se asigna un costo por apertura de *gap* y otro por extensión de *gap*. Así por ejemplo el costo total de un gap podría expresarse como,

$$G = g_0 + n \cdot g_e \tag{2.1}$$

donde G es el costo total mientras que g_0 denota apertura de gap, g_e extensión y n el número de posiciones del gap. Una combinación razonable y ampliamente utilizada de costos es $g_0 = -5$ y $g_e = -2$ [2].

Para evaluar pares de residuos, es de importancia distinguir entre nucleótidos y aminóacidos.

Los nucleótidos son la mayoría de las veces modelados como si tuvieran una probabilidad igual de mutar entre ellos. Esto es una simplificación, ya que las *transiciones* (cambios de purina a purina[3] o de pirimidina a pirimidina [4] i.e. $A \leftrightarrow G$ y $C \leftrightarrow T$) son mucho más frecuentes que las *transversiones* (cambios entre purinas y pirimidinas). Por simplicidad en este caso, solo se considera un costo cuando las bases son iguales y otro cuando son diferentes. Valores comunes para estos costos, son -3 para una diferencia y $+1$ para cuando son iguales.

En cambio, para los aminoácidos se toman en cuenta las diferentes probabilidades de que cambie un aminoácido por otro, ya que estas son mucho mas

[2] modelo de espacios afines
[3] Adenina y Guanina son purinas
[4] Timinia y Citosina son pirimidinas

diferentes y variadas. Esto debido a que son los codones, que especifican los aminoácidos, los que mutan, y existe redundancia en el sentido de que mas de un codon se traduce a un aminoácido. Esto tiene como consecuencia que para cambiar de un aminoácido a otro, se necesite un número variable de mutaciones, desde una hasta tres. Más aún, existe una gran diversidad de propiedades físico-químicas que tienen como resultado susbstituciones que alteran la estructura de una proteína en diferentes grados.

Para modelar estas diferencias entre las probabilidades de mutación entre aminoácidos, se han propuesto esquemas de matrices, donde se expresa la tasa de mutación asociada en una matriz. Los dos ejemplos mas visibles son las matrices PAM[14] y BLOSUM [24].

En resumen, ya sean nucleótidos o aminoácidos, para evaluar la similitud entre las secuencias alineadas, es común utilizar comparaciones entre unidades mínimas de un alineamiento (una posición alineada entre dos secuencias), lo cual de cierta manera deja a un lado uno de los criterios principales para valorar la calidad de un alineamiento, que es la parsimonia. Utilizar estas unidades mínimas de un alineamiento además implica que no se toma en cuenta el alineamiento como un todo directamente, si no como una construcción de alineamientos por pares, lo que posiblemente reduce la información disponible con la que se califica un alineamiento de múltiples secuencias. A estas situaciones se intenta dar una solución en el capítulo 3.

2.2.2. Alineamientos Óptimos

Alineamiento Global

Para realizar el alineamiento global óptimo de dos secuencias, el algoritmo más utilizado es el de *Needleman-Wunsch* [34], propuesto en 1970. Este algoritmo es una aplicación del método conocido en las ciencias de la computación como *programación dinámica*, siendo esta la primera aplicación de la programación dinámica a la comparación de secuencias biológicas.

El algoritmo hace uso de una *matriz de similitud*, en la que se va registrando la puntuación correspondiente, usando los modelos expuestos en la sección anterior, conforme se van alineando residuos. Cuando la matriz es llenada por completo, se obtiene la puntación del alineamiento óptimo, y con la información de la matriz se reconstruye dicho alineamiento.

Este algoritmo, debido a su naturaleza de programación dinámica, asegura regresar el alineamiento óptimo y tiene una complejidad computacional del orden $O(m \cdot n)$, donde m y n son las longitudes de las cadenas a alinear. El criterio de optimalidad se define según el esquema de evaluación que es utilizado (los costos por diferencias o igualdad entre residuos, además del modelo de gaps). De hecho, en teoría podría regresar un conjunto de alineamientos co-óptimos, pero en la mayoría de las implementaciones un solo elemento del conjunto de co-óptimos es entregado.

Alineamiento de Traslape

Para alinear dos segmentos que se traslapan solamente al inicio de uno y al final del otro, se utiliza el alineamiento de traslape. El algoritmo utilizado en este caso es simplemente una versión modificada del de Needleman-Wunsch, en el que los gaps iniciales y finales no son penalizados.

Este enfoque es la base de la técnica conocida como *secuenciación escopeta*[5] [3], utilizada ampliamente en la secuenciación de genomas completos.

Secuenciar un genoma, se puede realizar de dos maneras: (i) recorrer el genoma entero en forma secuencial, que es inherentemente un proceso lento debido a que como una sola lectura solo puede cubrir un número determinado de nucleótidos, son necesarias varias lecturas desde diferentes puntos de inicio específicos, o tomar un enfoque alternativo; (ii) *shotgun sequencing*, que consiste en leer fragmentos del genoma en forma aleatoria y en base a los traslapes entre ellos, reconstruir después el genoma entero.

Alineamiento Local

Dos proteínas o secuencias génicas de nucleótidos comúnmente comparten solamente *dominios homólogos* en lugar de tener homología en toda su extensión. En este caso se busca un alineamiento *local* en vez de global, como se muestra en la Figura 2.1. En el caso de un alineamiento local, lo que se busca en un par de subcadenas que alineadas entreguen una puntuación máxima. Y para el caso de alineamientos óptimos es el algoritmo de *Smith-Waterman* [42] el que resuelve el problema.

El algoritmo de Smith-Waterman, es también una variante del de Needleman-Wunsch, en el que con algunas modificaciones con respecto a las puntuaciones, se hacen visibles los alineamientos locales.

2.2.3. Alineamientos Heurísticos

El alineamiento óptimo de un par de secuencias, basado en el algoritmo de Needleman-Wunsch o alguna variante, tiene requerimientos de espacio y tiempo del orden de $O(m \cdot n)$, donde m y n son las longuitudes de las cadenas a alinear. Esto no se escala satisfactoriamente y los genomas tienden a tener desde miles hasta miles de millones de posiciones, lo que desalienta su uso. Existen mejoras a los algoritmos de programación dinámica con las cuales se reduce el requerimiento de espacio, pero para lograr mejoras en el tiempo de ejecución es necesario utilizar otras técnicas.

Alineamiento Global

El alineamiento global rápido, heurístico, es comúnmente usado para comparar genomas de organismos altamente relacionados. Esto es particularmente

[5] shotgun sequencing

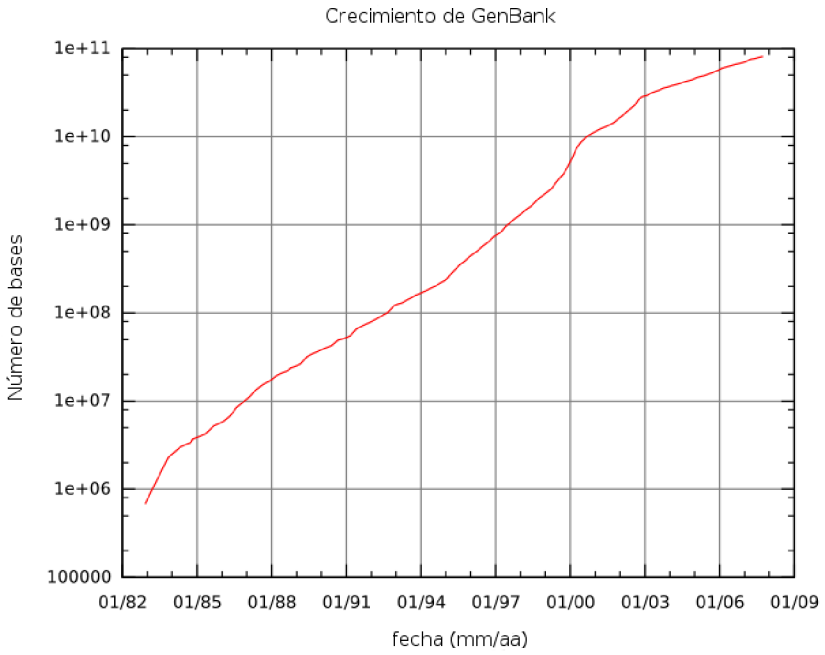


Figura 2.2: Crecimiento del GenBank. Adaptado de *Wikimedia Commons*, creado originalmente usando el documento que acompaño a la versión de GenBank [5] de octubre del 2007.

de interés en organismos donde la relación entre el genotipo y el fenotipo es comparativamente fácil de investigar, como es el caso de las bacterias.

Una de las herramientas para el análisis comparativo de genomas, por medio del alineamiento global de las secuencias que los representan es el software *MUMer* [15].

Alineamiento Local

En la actualidad existen bases de datos de secuencias genéticas y proteómicas que han crecido de manera exponencial, como es el caso de GenBank [5] y PDB [6]. El caso particular del crecimiento de GenBank se ilustra en la Figura 2.2.

Un enfoque común para utilizar estas bases de datos es buscar similitudes entre secuencias que se encuentran en la base de datos y secuencias recién obtenidas, con el objetivo de encontrar zonas homólogas a partir de las cuales, por ejemplo, se puedan inferir funciones similares para segmentos de la cadena recién secuenciada, o para la totalidad de esta. O incluso para verificar si la secuencia obtenida es potencialmente de la especie deseada y no de un contaminante de la muestra, como podría ser un hongo. Para realizar esta tarea el alineamiento local resulta una herramienta imprescindible, pero el volumen

impresionante de información disponible en las bases de datos, hace que la utilización de algoritmos óptimos de alineamiento local sea impráctico por sus demandas computacionales, por lo que han surgido algoritmos heurísticos para atacar el problema, de los cuales FASTA [38] y BLAST [1] son los más prominentes.

2.3. Alineamientos Múltiples

Uno de los principales motivos para alinear un par de secuencias, es que se desea inferir la función de una secuencia, con base en su similitud con otra. En el caso de alineamientos de múltiples secuencias, la motivación suele ser más diversa. De hecho, en el campo de la biología molecular, los alineamientos de múltiples secuencias son casi ubicuos. Son usados comúnmente en los siguientes escenarios:

- Encontrar [6] *motifs* en las secuencias, que caractericen a una familia de proteínas.
- Descubrir homologías entre una secuencia desconocida y familias de secuencias conocidas.
- Ayudar a la predicción de estructuras secundarias y terciarias.
- Ayudar en el diseño de *PCR primers*[7].
- Construir árboles filogenéticos.

Para alinear múltiples secuencias es posible extender los algoritmos óptimos para pares de secuencias que utilizan programación dinámica, que si bien aún conservan la cualidad de encontrar el mejor alineamiento existente según un criterio determinado, su costo computacional los hace inviables cuando se busca alinear conjuntos de decenas o centenas de secuencias, a pesar de que es posible utilizar algunas técnicas para optimizarlos.

Por esta razón, es más común que se utilicen técnicas heurísticas, que si bien no garantizan encontrar la mejor solución de todo el espacio de búsqueda, ofrecen resultados satisfactorios a la vez que son menos intensivos computacionalmente.

2.3.1. Evaluación de Alineamientos Múltiples

Para buscar alineamientos múltiples relevantes, es necesario primero definir un esquema de puntación. Uno de los mas populares para esta tarea es el método conocido como *suma de pares*[8], en el cual el alineamiento se expresa como una matriz de m por n donde m es el número de posiciones o columnas en el alineamiento y n el número de secuencias a alinear. La puntuación del alineamiento de múltiples secuencias, M, se define como

[6] motivos
[7] pequeños fragmentos de material genético que ayudan a la secuenciación.
[8] sum-of-pairs

$$M = \sum_{i=1}^{|A|} \sum_{j}^{n} \sum_{k<j} s(S_j[i], S_k[i]) \tag{2.2}$$

donde $s(S_j[i], S_k[i])$ es la puntuación (entero positivo, que se define en base a matrices como PAM[14] y BLOSUM [24], o como un valor para posiciones idénticas y otro para diferentes) de la posición i de la secuencia j alineada con la posición i de la secuencia k (están alineadas las bases o gaps en una misma posición). Al alineamiento de dos símbolos de *gap* ("−") le es asignada una puntación de cero.

Una variante de *suma de pares* es la *suma de pares pesada* o *ponderada* [9] en la cual se agrega un peso específico para cada par de secuencias dependiendo de su distancia evolutiva, con el objetivo de darle mas peso a los alineamientos de secuencias más próximas entre si. La ecuación para *weighted sum-of-pairs* sería la siguiente,

$$M = \sum_{i=1}^{|A|} \sum_{j}^{n} \sum_{k<j} w_{jk} \cdot s(S_j[i], S_k[i]) \tag{2.3}$$

donde w_{jk} es el peso asociado al par de secuencias j y k dependiento de su distancia evolutiva.

Calcular la distancia evolutiva entre dos secuencias nos lleva a otro problema, esta información muy pocas veces se encuentra disponible antes de realizar el alineamiento (el estudio filogenético es generalmente un paso subsecuente al alineamiento de las secuencias), por lo que muchas veces ese valor solo es una aproximación en base a la similitud entre las secuencias.

Otras variaciones importantes a *suma de pares* y *suma de pares pesada* son:

- La utilización de matrices de substitución, tales como BLOSUM y PAM, para determinar las puntuaciones de alineación de residuos $s(S_j[i], S_k[i])$.
- Utilizar diferentes esquemas para la puntuación de los *gaps*, como el *affine gap model* u otros esquemas.

2.3.2. Alineamientos Óptimos

Los algoritmos para realizar alineamientos globales de un par de secuencias pueden ser extendidos para manejar un número mayor de estas. Como se mencionó en la sección 2.2.2, para llevar acabo un alineamiento por medio del algoritmo de Needleman-Wunsch se utiliza una matriz bidimensional. En el caso de la extensión de este algoritmo para múltiples secuencias, esta matriz bidemensional se convierte en una matriz n-dimensional lo cual repercute en que el algoritmo tenga una complejidad de $O(2^n \prod_{i=1}^{n} |S_i|)$ haciendo a este algoritmo y a sus derivados poco viables de ser utilizados en los escenarios comunes de alineamiento múltiple. Para sortear este obstáculo se han creado algoritmos

[9] weighted sum-of-pairs

heurísticos y son estos los ampliamente difundidos y usados por la comunidad interesada en el problema.

2.3.3. Alineamientos Heurísticos

La mayoría de los algoritmos heurísticos de alineamiento múltiple realizan la tarea básicamente haciendo alineamientos entre pares de secuencias. Con la particularidad de que para que este proceso sea exitoso, es necesario hacer los alineamientos por pares en el orden correspondiente a las relaciones evolutivas (cercanía) entre las secuencias a ser alineadas. Posiblemente la técnica heurística más utilizada para realizar alineamientos múltiples es el *alineamiento progresivo*. Este método fue desarrollado en 1987 por Feng y Doolittle [17] y es la base para programas de alineamiento como *CLUSTALW* [39].

Cuando Feng y Doolittle publicaron su propuesta, el método estándar para construir alineamientos de múltiples secuencias era generar todos los alineamientos por pares y después unirlos manualmente considerando las relaciones evolutivas entre las secuencias. Los autores comentaron lo siguiente: "Nos parece poco sensato que un *gap* en una secuencia sea descartado en el alineameinto de dos secuencias relacionadas cercanamente solamente porque mejoraría el alineamiento con otra secuencia con la que guarda menos relación".

Como secuencias similares pueden ser alineadas con mayor éxito que secuencias más distantes, al alineamiento de secuencias similares debe tener más peso dentro del esquema de alineamiento general. Este mayor peso se logra alineando las secuencias progresivamente, comenzando con el par mas similar y seguir la regla de que una vez insertado un *gap*, este se conservará en los subsecuentes pasos.

El enfoque de alineamientos progresivos ha seguido evolucionando y en la actualidad existen programas de alineamiento múltiple que sobrepasan las capacidades de *CLUSTALW*, en términos de velocidad y posiblemente calidad de los alineamientos. Un ejemplo prominente es el programa *MUSCLE* propuesto por Edgar, R. C.[16].

Existen otros enfoques heurísticos para el problema, como los *Hidden Markov Models*[10] que son modelos probabilísticos que asignan *probabilidades*[11] para todas las posibles combinaciones de *gaps* y alineamiento de posiciones, para después determinar el alineamiento más probable o el conjunto de alineamientos más probables.

Así mismo, el *recocido simulado*[12] [26] y técnicas de cómputo evolutivo también han sido utilizadas para atacar el problema de alineamiento. Estas son técnicas de optimización que requieren de una función objetivo, la cual generalmente es alguna variante de *suma de pares*. La utilización del cómputo evolutivo para alineamiento de secuencias se discute en la siguiente sección.

[10] Modelo oculto de Markov
[11] likelihoods
[12] simulated annealing

2.4. Cómputo Evolutivo y Alineamiento de Secuencias

Las técnicas de cómputo evolutivo han sido aplicadas para atacar el problema del alineamiento de múltiples secuencias, incluyendo a la programación evolutiva [11] [8] y a los algoritmos genéticos [27] [35] [37].

Una de las principales ventajas de utilizar el cómputo evolutivo, es que este permite una conveniente separación entre el proceso de optimización y el criterio de evaluación (conocido como la *función objetivo*). Es la función objetivo la que define el propósito de cualquier procedimento de optimización. Es en este criterio donde se busca abstraer parte del conocimiento que se tiene sobre el área del problema que se está atacando. Y en el caso del alineamiento de múltiples secuencias la construcción o elección de la función objetivo reflejan los conceptos en la biología que definen un mejor alineamiento y que se desean proyectar en la búsqueda de los mismos.

2.4.1. SAGA

SAGA (*Alineamiento de Secuencias por Algoritmo Genético*[13] por sus siglas en inglés) es un método propuesto por Notredame y Higgins [35] que utiliza una variación del algoritmo genético simple de Goldberg [22].

Los algoritmos genéticos son una técnica de optimización, en la cual se tiene una *población* de soluciones (individuos) que *evoluciona* a través de *generaciones* dando preferencia a las soluciones más *aptas*, es decir con un mejor valor de la función objetivo a optimizar. Las soluciones evolucionan por un proceso de *selección*, *mutación* y *cruza* para variarlas y de esa manera llegar a soluciones mejores.

En SAGA cada individuo es un alineamiento múltiple. La representación usada para los individuos es simplemente una matriz bidimensional donde cada línea es una secuencia del alineamiento, y cada celda es un residuo o un gap. La población tiene un tamaño constante y no contiene duplicados (i.e., individuos idénticos). En la Tabla 2.3 se presenta el pseudocódigo general de SAGA.

Inicialización

En SAGA, la generación inicial consiste de 100 alineamientos múltiples generados aleatoriamente, insertando gaps al inicio y al final de cada secuencia. Estos alineamientos iniciales son de menos del doble de tamaño que la secuencia más larga del conjunto. Para crear estos individuos, se inserta un gap inicial de tamaño aleatorio (entre 0 y el tamaño de la secuencia más larga), y después se rellena la matriz con signos de nulo para que todas las secuencias tengan la misma longitud.

[13] Sequence Alignment by Genetic Algorithm

Inicialización:	1.	*crear G_0, una población inicial aleatoria*
Selección:	2.	*evaluar la población de la generación n (G_n)*
	3.	*si la población no cambió entonces ir a* **Fin**
	4.	*seleccionar los individuos a reemplazar*
	5.	*evaluar los descendientes esperados*
Variación:	6.	*seleccionar los padres desde (G_n)*
	7.	*seleccionar un operador*
	8.	*generar los descendienes*
	9.	*guardar o descartar a los nuevos descendientes en G_{n+1}*
	10.	*ir al paso 6 hasta que G_{n+1} sea compleatada*
	11.	$n = n + 1$
	12.	*ir al paso 2*
Fin:	13.	*fin*

Tabla 2.3: Pseudocódigo general de SAGA [20].

Evaluación

La aptitud se obtiene evaluando cada alineamiento usando la función objetivo escogida. SAGA ha sido probado usando dos funciones objetivo, la suma de pares (*sum-of-pairs*) y otra función propuesta por el mismo Notredame, COFFEE [36]. Esta última utiliza una librería de alineamientos por pares, como referencia para evaluar los pares existentes en los alineamientos múltiples.

Para las dos funciones objetivo, a alineamientos mejores se le asignan puntuaciones más altas. Para minimizar errores de muestreo, las puntuaciones originales son convertidas a un valor normalizado conocido como *expected offspring*[14] (EO), que en este caso se deriva estocásticamente usando un método predefinido llamado muestreo estocástico sin remplazo (*stochastic sampling without replacement*) [22].

Sólo la mitad de los individuos, los que tienen la aptitud más baja, es remplazada con nuevos individuos, la otra mitad persiste sin cambio hacia la siguiente generación.

Reproducción, Varación y Terminación

Durante la fase de variación es cuando los nuevos individuos son generados. El valor de *expected offspring* es utilizado como la probabilidad de cada individuo de ser escogido como padre. La selección se realiza utilizando un método de ruleta, y el valor de *expected offspring* es decrementado en una unidad cada que el individuo es elegido como padre.

Un operador de variación es elegido entonces, y aplicado al individuo padre o individuos padres (dependiendo del operador). Como opciones se tienen dos ope-

[14] descendencia esperada

radores de cruza, que requieren de dos padres, y veinte operadores de mutación, que solo necesitan de un padre. Cada uno tiene su propia probabilidad de uso. Debido a que no se admiten individuos duplicados en la población, un nuevo individuo es aceptado solo si es diferente a los existentes. La fase de variación se completa cuando la nueva generación ha sido completamente generada. Y se procede a la siguiente generación, hasta que se cumple el criterio de paro, que para SAGA es cuando han pasado 100 generaciones y no se ha conseguido mejorar la puntuación.

Adaptación de Operadores

Al crear nuevos individuos el operador de variación elegido tiene mucha influencia, por lo que resulta atractivo adaptar la probabilidad de uso de los operadores y de esa manera favorecer la utilización de los operadores que tengan un mejor impacto en la puntuación de los individuos. Inicialmente, como no es posible saber que operadores serán mas benéficos, a todos se les asigna la misma probabilidad. Al pasar las generaciones las probabilidades son ajustadas para reflejar la eficacia de los operadores. Cuando un operador mejora la puntuación de un individuo, este comparte el crédito con los operadores que han actuado con anterioridad en el individuo. El crédito por las mejoras de cada operador se acumula por algunas generaciones para después cada cierto tiempo modificar las probabilidades de los operadores de acuerdo a los beneficios que causó.

2.4.2. Nuevas perspectivas

En el siguiente capítulo se presenta una propuesta de función objetivo para evaluar la calidad de un alineamiento de múltiples secuencias genéticas, la cual considera conceptos que no han sido completamente incorporados en otros enfoques de evaluación y que aportan información e indicios sobre como evaluar los alineamientos. Estos conceptos son específicamente, la parsimonia y la consideración de todas las secuencias que conforman el alineamiento a la vez.

Usando esta nueva función de evaluación propuesta, en el capítulo 4 se describe un algoritmo genético el cual es capaz de refinar alineamientos de múltiples secuencias genéticas realizados previamente por otro programa de alineamiento progresivo, haciendo así uso de algoritmos ya existentes e incluyendo las nuevas aportaciones en la búsqueda de mejores resultados.

Capítulo 3

GLOCSA

Como se vio en la sección 2.3.1, existen formas para evaluar la calidad de un alineamiento de múltiples secuencias, basadas principalmente en la comparación de pares de residuos en cierta posición del alineamiento, no de la posición entera. Es por eso que se propone una nueva medida en la cual se considera la posición en su totalidad para darle una calificación a esa columna y usando esa puntuación, evaluar el alineamiento completo. Bajo esta perspectiva se concibió GLOCSA (Criterio Global para Alineamiento de Secuencias por sus siglas en inglés[1]) para evaluar alineamientos de múltiples secuencias de ADN. Aunque es posible extender el concepto para poder evaluar alineamientos de ARN y proteínas, en este trabajo solamente se desarrolló para tomar en cuenta secuencias de ADN.

GLOCSA, en sí, es la unión de tres criterios independientes en un polinomio de primer grado, con pesos asociados. Estos tres criterios son:

- Homogeneidad de Columna.
- Concentración de Gaps.
- Incremento de Columnas.

La forma general se muestra a continuación,

$$GLOCSA = w_{mch}MCH + w_{gc}GC + w_{ci}CI \tag{3.1}$$

donde MCH es la homogeneidad promedio de columnas, GC es la concentración de gaps y CI es el incremento de columnas. Además w_{mch}, w_{gc} y w_{ci} son sus pesos asociados respectivamente, cuyos valores por default propuestos se muestran en la Tabla 3.1. Estos valores por default se obtuvieron al realizar pruebas empíricas sobre conjuntos de secuencias especialmente creados para calibrar la función. Esta función entrega una puntación mayor para alineamientos considerados de mejor calidad.

[1] Global Criterion for Sequence Alignment

Tabla 3.1: Pesos Default para GLOCSA

w_{mch}	$= 1000$
w_{gc}	$= 20$
w_{ci}	$= -20$

3.1. Homogeneidad de Columnas

El criterio de *homegeneidad de columnas* busca premiar las columnas[2] donde exista una menor diversidad de bases, en las posiciones donde exista mayor consistencia en el alineamiento. Esta medida da una mayor calificación a las columnas más uniformes, más homogéneas y una menor a las que exponen mayor diversidad, las más heterogéneas.

Para calcular la homogeneidad de una columna, se comienza contando las ocurrencias de cada posible base ($[A, C, G, T]$) y cuantas codificaciones de gap ($-$) existen en la columna o posición de interés.

Un punto importante es que en la codificación de secuencias de ADN, es muy frecuente encontrar polimorfismos, que son codificaciones que indican que en esa posición se tiene incertidumbre sobre la base que representa, que puede ser una entre dos o cuatro bases posibles. Incluso se puede desconocer si existe una base en esa posición, considerándose entonces como cualquier base o gap. En la Tabla 3.2 se muestran las codificaciones admitidas, estas codificaciones son estándar en el ámbito y son ampliamente usadas, como por ejemplo en el formato de archivo FASTA [38] para secuencias genéticas.

Cuando al calcular la homogeneidad de columna se encuentra un polimorfismo, se reparte equitativamente la unidad a contar entre las posibles bases que representa. De esta manera, por ejemplo, si se encuentra una R en una columna, ésta contará $1/2$ para G y $1/2$ para A, o si se encuentra una B esta contará $1/3$ para G y $1/3$ para T y $1/3$ para C.

La información de los pesos correspondientes a cada base o gap dependiendo de la codificación puede ser expresada en la matriz que se muestra en la Tabla 3.3, donde cada columna encabezada por una codificación contiene los pesos correspondientes a las 4 bases y el gap ($[A, C, G, T, -]$) indexados por b.

En el caso particular de la codificación ? que representa cualquier base o gap, es decir ausencia de información sobre esa posición, al evaluar una columna con una secuencia con esta codificación simplemente no se toma en cuenta esa secuencia para el cálculo de la homogeneidad. Por ejemplo, en la Tabla 3.4 se observa una columna donde la última secuencia presenta la codificación ?, como esta no se toma en cuenta, la columna se evalua como si la secuencia no estuviese ahí, como si fuera la columna mostrada en la Tabla 3.5.

Las cuentas *pesadas* para cada base y el gap son acumuladas en $wc_{jb} \quad \forall \quad 0 \leq b \leq 4$, donde b es el índice con el cual se indica la base o gap (como se puede observar en la Tabla 3.6) que se está contando y j denota la columna j-ésima

[2] conjunto de codificaciones de base o gap que se encuetran en una misma posición para cada una de las secuencias de un alineamiento

Tabla 3.2: Codificaciones admitidas para Ácidos Nucléicos

A	Adenina
C	Citosina
G	Guanina
T	Timina
R	G ó A
Y	T ó C
K	G ó T
M	A ó C
S	G ó C
W	A ó T
B	G ó T ó C
D	G ó A ó T
H	A ó C ó T
V	G ó C ó A
N	cualquier base
-	gap
?	cualquier base o gap

Tabla 3.3: Matriz de pesos para contar bases

b		A	C	G	T	R	Y	K	M	S	W	B	D	H	V	N	–
0	A	1	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	0	$\frac{1}{2}$	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{4}$	0
1	C	0	1	0	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{4}$	0
2	G	0	0	1	0	$\frac{1}{2}$	0	$\frac{1}{2}$	0	$\frac{1}{2}$	0	$\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{4}$	0
3	T	0	0	0	1	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{1}{4}$	0
4	–	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

de alineamiento que se está evaluando.

Esto puede también ser expresado de la siguiente manera,

$$wc_{jb} = \sum_i B_w\left(b, am(i,j)\right) \tag{3.2}$$

donde $am(i,j)$ es la codificación existente en el alineamiento en la posición o columna j y la secuencia i, mientras que $B_w(b, am(i,j))$ es el peso asociado a la base b de la codificación $am(i,j)$, pesos que se indican en la Tabla 3.3.

Una columna de ejemplo, de un alineamiento hipotético de 10 secuencias, junto con su respectivo conteo pesado se puede observar en las Tablas 3.7 y 3.8. Al final, para A se acumulan $\frac{10}{3}$, 3 unidades de las primeras 3 secuencias y $\frac{1}{3}$ de la secuencia 7 (D puede ser G, A ó T, por lo que aporta $\frac{1}{3}$ para cada una); para C es $\frac{1}{2}$ proveniente de la secuencia 6 (S puede ser G ó C, por lo que cuenta para cada una $\frac{1}{2}$); para G se tiene un total de $\frac{17}{6}$, 2 unidades de las secuencias 3 y 4, mas $\frac{1}{2}$ de la secuencia 6 y $\frac{1}{3}$ de la secuencia 7; para T se obtienen $\frac{4}{3}$, 1 unidad de la secuencia 5 y $\frac{1}{3}$ de la secuencia 7; finalmente para – se suman 2 por las

Tabla 3.4: Columna de ejemplo con una codificación ?

sec0	A
sec1	A
sec2	-
sec3	?

Tabla 3.5: Columna equivalente sin ?

sec0	A
sec1	A
sec2	-

Tabla 3.6: Indexación de las cuatro bases y el gap

b	base o gap
0	A
1	C
2	G
3	T
4	$-$

codificaciones de gap de las dos últimas secuencias.

Una vez realizado el conteo, la homogeneidad de una columna se calcula de la siguiente manera,

$$CH_j = \frac{\sum_{b=0}^{3} \left(wc_{jb}\right)^2}{\left(\sum_{b=0}^{4} wc_{jb}\right)^2} \tag{3.3}$$

Es de notarse, que en la parte superior de la fracción descrita anteriormente la sumatoria sólo incluye las 4 bases principales $[A, C, G, T]$ (ya que b solamente va de 0 a 3) mientras que en la parte inferior si es considerado el gap $-$ junto con las bases (b va de 0 a 4). Esto con la intención de desalentar la inserción de gaps en los alineamientos, siguiendo el raciocinio de que como no son contados en el numerador, su existencia en una columna disminuye el valor de homogeneidad.

Cuando una columna contiene solamente codificaciones "$-$", el valor de homegeneidad de columna es cero ($CH_j = 0$).

Además, a diferencia del criterio de *concentración de gaps* (que se describe en la sección 3.2), para el cálculo de la homogeneidad de columnas sí se toman en cuenta las codificaciones "$-$" que conforman los gaps inciales y finales (los que se encuentran al inicio o al final de una secuencia, respectivamente). Si éstos no se tomarán en cuenta para el cálculo, es decir que sólo se consideraran las secuencias que en la columna que se está trabajando no presenten un gap incial o final como las únicas existentes en el alineamiento, no se penalizaría apropiadamente la inserción de gaps iniciales (los gaps finales solo son una consecuencia de la disparidad de la longuitud de las secuencias), lo que nos llevaría a alineamientos con muchas codificaciones "$-$", los cuales no serían óptimos.

Para la columna de ejemplo de la Tabla 3.7 (asumiendo que es la j-ésima columna), el cálculo de la homogeneidad de columnas sería el siguiente,

Tabla 3.7: Columna de ejemplo

sec0	A
sec1	A
sec2	A
sec3	G
sec4	G
sec5	T
sec6	S
sec7	D
sec8	-
sec9	-
CH	0.2117

Tabla 3.8: Conteo de la columna de ejemplo

	A	C	G	T	−
conteo	$\frac{10}{3}$	$\frac{1}{2}$	$\frac{17}{6}$	$\frac{4}{3}$	2

$$CH_j = \frac{\left(\frac{10}{3}\right)^2 + \left(\frac{1}{2}\right)^2 + \left(\frac{17}{6}\right)^2 + \left(\frac{4}{3}\right)^2}{\left(\frac{10}{3} + \frac{1}{2} + \frac{17}{6} + \frac{4}{3} + 2\right)^2} = 0.2117$$

Después de obtener el valor de homegeneidad de cada columna simplemente se calcula el valor medio para todas las columnas del alineamiento,

$$MCH_j = \frac{1}{N}\left(\sum_{j=0}^{N-1} CH_j\right) \tag{3.4}$$

De esta manera, mediante la incorporación del valor medio de la homogeneindad de columnas a GLOCSA, se busca penalizar con un valor bajo a aquellas columnas con mucha diversidad, dando a su vez, un valor alto a las columnas más homogéneas (como puede observarse en la Tabla 3.9), favoreciendo de esta manera a matrices más alineadas.

3.2. Concentración de Gaps

La *concentración de gaps* es un criterio propuesto para reflejar el grado de agrupación de las codificaciones individuales de gap en una secuencia, cada −, en bloques a lo largo de todo el alineamiento. Desde un punto de vista (biológicamente) evolutivo es más parsimonioso encontrar, por ejemplo, un gap de 5 posiciones, que 5 gaps de una sola posición, ya que cada bloque de codificaciones de gap representan un evento *indel*. Es por eso que se incluye este criterio en GLOCSA, para premiar a los alineamientos con una concentración mayor, que implican un número menor de eventos necesarios para explicarlo.

Mediante la siguiente ecuación se realiza el cálculo de la concentración de gaps,

Tabla 3.9: Ejemplo de evaluaciones para homogeneidad de columnas

	columna												
	0	1	2	3	4	5	6	7	8	9	10	11	12
sec0	A	A	A	A	A	A	A	A	A	A	–	A	A
sec1	A	A	A	A	A	A	A	A	A	A	–	–	G
sec2	A	A	A	A	A	A	A	A	A	G	–	–	–
sec3	A	A	A	A	A	A	A	A	A	G	–	–	–
sec4	A	A	A	A	A	A	A	A	G	T	–	–	–
sec5	A	A	A	A	A	A	A	A	G	T	–	–	–
sec6	A	A	A	A	A	A	A	G	T	T	–	–	–
sec7	A	A	A	A	A	A	G	G	T	C	–	–	–
sec8	A	A	–	A	G	G	T	T	C	C	–	–	–
sec9	A	–	–	G	G	T	C	T	C	C	–	–	–
CH	1.00	0.81	0.64	0.82	0.68	0.66	0.52	0.44	0.28	0.26	0.00	0.01	0.02

$$GC = \frac{\overline{S_{GB}}}{GP} \tag{3.5}$$

donde S_{GB} es el tamaño promedio de un bloque de codificaciones de gap contiguas y GP es el total de codificaciones de gap individuales en todo el alineamiento.

Es de notar que los gaps iniciales (los que se encuentran antes de cualquier codificación de bases en una secuencia, i.e. al inicio de ella) y los gap finales (los que se encuentran después de todas las codificaciones para alguna base, i.e. al final de la secuencia) no son tomados en cuenta para el cómputo de la concentración de gaps.

Otra consideración especial se toma cuando no existen codificaciones de gap en el alineamiento ($GP = 0$), en este caso la concentración de gaps se hace cero ($GC = 0$).

En las Tablas 3.10, 3.11 y 3.12 se muestran tres alineamientos de un conjunto hipotético de secuencias, los tres tienen el mismo número de codificaciones de gap, pero 3.10 las tiene agrupadas en 3 bloques, 3.11 en 2 bloques y 3.12 en un sólo bloque, lo cual se refleja en su valor de concentración de gaps.

3.3. Incremento de Columnas

El *incremento en columnas* busca penalizar el crecimiento del alineamiento, resultado indirecto de la inserción de gaps. Si bien muchas veces el aumento en el número de columas es inevitable cuando se alinea un conjunto de secuencias, es preferible que este aumento sea lo más pequeño posible.

Para calcular este incremento se utiliza la siguiente ecuación,

Tabla 3.10: Alineamiento para ilustrar el criterio de Concentración de Gaps. $GC = 0.3\bar{3}$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
sec0	A	A	A	A	G	G	C	A	T	C	A	T	C	A	T	C	A	G	G	A	A	A	A
sec1	A	A	A	A	G	G	-	-	-	C	-	-	-	A	-	-	-	G	G	A	A	A	A

Tabla 3.11: Alineamiento para ilustrar el criterio de Concentración de Gaps. $GC = 0.50$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
sec0	A	A	A	A	G	G	C	A	T	C	A	T	C	A	T	C	A	G	G	A	A	A	A
sec1	A	A	A	A	G	G	-	-	-	-	-	-	C	A	-	-	-	G	G	A	A	A	A

Tabla 3.12: Alineamiento para ilustrar el criterio de Concentración de Gaps. $GC = 1.0$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
sec0	A	A	A	A	G	G	C	A	T	C	A	T	C	A	T	C	A	G	G	A	A	A	A
sec1	A	A	A	A	G	G	C	-	-	-	-	-	-	-	-	-	A	G	G	A	A	A	A

Tabla 3.13: Alineamiento para ilustrar el criterio de Incremento en Columnas. En este caso, se mantiene el número de columnas al alinear, no se da ningún incremento. $CI = 0$

	0	1	2	3	4	5	6	7	8
sec0	A	T	C	A	T	C	A	T	C
sec1	A	T	C	A	T	C	A	T	C
sec2	A	T	C	A	T	C	A	T	C

Tabla 3.14: Alineamiento para ilustrar el criterio de Incremento en Columnas. Aquí se incrementó en 6 el número de columnas después de alinear. $CI = 0.6\bar{6}$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
sec0	A	T	C	A	T	C	-	-	-	A	T	C	-	-	-
sec1	A	T	C	-	-	-	A	T	C	A	T	C	-	-	-
sec2	A	T	C	-	-	-	-	-	-	A	T	C	A	T	C

$$CI = \frac{C}{C_0} - 1 \tag{3.6}$$

donde, C es el número de columnas en el alineamiento que se está evaluando y C_0 es el número de columnas de la matriz cuando el conjunto de secuencias no se encuentra alineado, es decir cuando no tiene codificaciones de gap (que es equivalente al tamaño de la secuencia más larga sin tomar en cuenta los "−"). Recuérdese que en GLOCSA, el incremento de columnas tiene asociado un peso negativo, por lo que se penalizan los incrementos mayores.

En las Tablas 3.13 y 3.14 se da un ejemplo de un conjunto hipotético de secuencias para los que se muestran dos diferentes alineamientos con diferentes incrementos en columna.

3.4. GLOCSA y eventos evolutivos

A través de los criterios incluidos en GLOCSA, se busca indirectamente dar un indicador del número de eventos necesarios para explicar un alineamiento[3], entendiendo que un alineamiento es mejor cuando implica una menor cantidad de eventos, pero sin caer en una simplificación excesiva como sería el caso de sólo suponer *indels* que tengan como resultado un alineamiento donde existan posiciones con sólo una secuencia con codificación y las demás con *gaps*, como se puede observar en la Tabla 3.14.

[3]Esto se pude observar en la aplicación web que se detalla en la sección A.1, donde se muestra el número de sinapomorfías potenciales para el alineamiento que se evalua.

Capítulo 4

Algoritmos Genéticos y GLOCSA

4.1. Generalidades sobre Algoritmos Genéticos

John Holland y sus estudiantes en la Universidad de Michigan fueron quienes a través de su trabajo, que se originó del estudio de autómatas celulares, introdujeron formalmente y popularizaron el concepto de algoritmos genéticos en los años 1970 [25] . Aunque con anterioridad se habían realizado trabajos relacionados, muchos de los cuales fueron recopilados for David B. Fogel en *The Fossil Record* [18].

Los algorimos genéticos son un método de optimización que junto con otros, como la programación genética [30], la programación evolutiva [21] y las estrategias evolutivas [40] forman parte de lo que se conoce como *algoritmos evolutivos*, que a su vez se consideran dentro del área más general conocida como *cómputo evolutivo*.

Los algoritmos evolutivos en general, y por lo tanto los algoritmos genéticos, se inspiran en una metáfora del proceso evolutivo que se da en los seres vivos, i.e. en la selección natural de los invidudos más aptos, la existencia de un conjunto de información genética (genotipo) que define las características de un individuo (fenotipo) y la recombinación y mutación de la información genética que genera nuevos genotipos (y en consecuencia nuevos fenotipos) que competirán por ser los más aptos.

Siguiendo la analogía para los algoritmos genéticos, las posibles soluciones definidas por el espacio de búsqueda que se intenta optimizar son potenciales individuos, o más específicamente genotipos, de los cuales se tiene una población que evoluciona a través de generaciones o iteraciones de ciclos de selección de los más aptos (con base en un criterio de optimalidad sobre el fenotipo, i.e la función objetivo) a partir de los cuales se obtiene la siguiente generación mediante un proceso que incluye la conservación, variación y recombinación de los individuos (operadores del algoritmo genético, e.g. mutación, cruza, elitismo y otros).

En el proceso de selección de los más aptos y en los procesos que obtienen la nueva generación de la población de individuos que está evolucionando, existe un componente de aleatoriedad. En la selección por ejemplo, es común solamente asignar probabilidades más altas de ser seleccionados a los individuos más aptos, y para crear nuevos individuos se pueden combinar dos individuos existentes de manera aleatoria (operador de recombinación o cruza) o simplemente realizar un cambio aleatorio en el individuo (operador de mutación). Pero a pesar de esta aleatoridad en el proceso de búsqueda, además de que, en primera instancia, no se incorpora directamente conocimiento del problema sobre como explorar el espacio de búsqueda de manera eficiente, los algoritmos genéticos están lejos de llevar a cabo una búsqueda aleatoria. De hecho son considerados como un método de optimización robusto, con relativa eficiencia y eficacia en problemas variados, ventajas que se pueden incrementar al incorporar conocimiento específico sobre el domino del problema a los operadores del algoritmo genético.

Los algoritmos genéticos en general son computacionalmente hablando bastante sencillos, pero sin embargo tienen un gran potencial para encontrar buenas soluciones en un espacio de búsqueda complejo [22]. Además, estos no imponen restricciones a los espacios de búsqueda que pueden explorar, como otros métodos de optimización que requieren que estos sean continuos, que sean derivables, unimodales y otras restricciones.

La validez de este método de optimización ha sido probada empíricamente en diversos ámbitos e.g. problemas de negocio, aplicaciones en la ciencia y en la ingeniería [10] [28] [19] y se han realizados diversos esfuerzos para darle fundamentos teóricos [25] [22]. Pero la investigación de su teoría subyacente es aún objeto de un activo interés y desarrollo en busca de lograr un consenso de como es que funcionan (es decir se adaptan para encontrar mejores soluciones).

La estructura general de un algoritmo génetico se muestra a continuación.

1. *Generar población inicial*
2. *Evaluar la población*
3. *Generar una nueva población a través de los procesos de preservación (selección, elitismo) y los procesos de variación del algoritmo (operadores de mutación, cruza y otros)*
4. *Volver a (2) mientras no se cumpla la condición de paro impuesta*

En el paso (1) se generan por primera vez los individuos que conformarán la población del algoritmo genético, cada individuo es una solución posible dentro del espacio de búsqueda que define la representación elegida. Esta generación inicial muchas veces es aleatoria, pero bien puede ser inicializada a partir de ciertos valores para, por ejemplo, comenzar la búsqueda desde una región más factible. Evaluar la población en el paso (2) se lleva a cabo usando la función objetivo definida para determinar la calidad de una solución, con base en el valor de aptitud que se le asigna. La función objetivo es básicamente la que dirige el proceso de búsqueda de las soluciones, al definir cuales son las regiones más factibles del espacio de búsqueda y que soluciones son las mejores. En el paso (3) se crea una nueva población, la siguiente generación de esta, a partir de un

proceso que involucra el equilibro entre explotación (preservación de las buenas soluciones encontradas y su selección para apartir de ellas generar nuevas) y exploración (variación de las soluciones encontradas para intentar nuevas). En el último paso, simplemente se revisa si se cumple el criterio definido para finalizar el algoritmo, que la mayoría de las ocasiones es un número determinado de generaciones, pero bien puede ser si el valor de la función objetivo alcanzado por algún individuo supera un valor definido.

En resumen se pueden señalar tres componentes principales para los algoritmos genéticos, i.e. la representación de las soluciones, la función objetivo, y los operadores que actúan sobre los individuos de la población.

La representación de las posibles soluciones es de gran trascendencia para el proceso de optimización ya que apartir de ésta se define el espacio de búsqueda que será explorado. Comúnmente las soluciones se representan como cadenas (en analogía a cromosomas) binarias, de enteros o de números reales, aunque de ninguna manera se restringe a éstas. Pueden ser usadas estructuras de datos más complejas, como en el caso de la programación genética (un subconjunto de los algoritmos genéticos) que se usan árboles para representar a las soluciones.

Estrechamente ligado a la representación está la función objetivo, la cual evalua las soluciones en la representación elegida, para asignarles un puntaje representativo de su calidad. El proceso por medio del cual se asigna el valor de la función objetivo depende totalmente del problema a optimizar, ya que es la definición del problema la que nos indica cuales son sus (mejores) soluciones.

Mientras que la función objetivo guía el proceso de búsqueda, son los operadores la parte fundamental que permite llevar a cabo la búsqueda. Por la misma naturaleza de su proceso de búsqueda, es necesario llegar a un equilibrio entre exploración y explotación, es decir entre hacer modificaciones de gran impacto para sondear regiones inexploradas del espacio de búsqueda, y conservar (o modificar solo ligeramente, como haciendo una búsqueda local) las mejores soluciones encontradas hasta el momento. Es precisamente este equilibrio y su relación con el tipo de operadores y sus parámetros asociados un aspecto fundamental de los algoritmos genéticos, el cual también es un campo activo de investigación.

Con respecto a la conservación se pueden considerar a los operadores de *Selección* y *Elitismo*. La selección es el proceso mediante el cual se eligen los individuos para pasarlos a la siguiente generación o a los otros operadores; esta elección se hace en base al valor de la función objetivo, favoreciendo a las mejores soluciones. El elitismo es asegurar que el mejor o un número determinado de los mejores individuos pasen a la siguiente generación, con el propósito de conservar siempre las mejores soluciones.

Para explorar nuevas soluciones los operadores más comunes son los de *Mutación* y *Cruza* (también llamado *Recombinación*). La mutación es introducir, con cierta probabilidad, cambios aleatorios en los individuos. El proceso de cruza consiste en mezclar el genotipo de dos indiviudos para crear nuevas soluciones que sean recombinaciones de las soluciones ya existentes. La cruza también se aplica con cierta probabilidad, la cual es un parámetro del algoritmo. Otros operadores son posibles, especialmente si se incorpora conocimiento específico

del problema a su diseño, con el objetivo de eficientar el proceso de búsqueda.

Finalmente, es valioso notar que es en la población, en el conjunto de individuos que representan las soluciones que se están explorando en cada generación, donde se condensa indirectamente la información que el algoritmo genético tiene sobre el espacio de búsqueda. Ya que en la población, conforme pasan las generaciones, se reflejan las zonas con mejores soluciones que el algoritmo ha encontrado.

4.2. GGGA - Algoritmo Genético Guidado por GLOCSA

Teniendo a *GLOCSA* como un criterio para evaluar la calidad de un alineamiento de secuencias genéticas se implementó GGGA (*GLOCSA Guided Genetic Algorithm*[1]), un algoritmo genético que utiliza como función objetivo a GLOCSA, buscando mejorar alineamientos en base a este criterio usando técnicas del cómputo evolutivo.

Se planteó una representación específica para los alineamientos, y un operador de mutación con cinco suboperadores acordes a la representación y al problema en cuestión. Después de considerar el uso de un operador de cruza que recombinaba los alineamientos intercambiando secuencias (taxas) entre ellos, éste fue descartado por resultar demasiado destructivo, dejando únicamente a la mutación para introducir variación a la población. Se incluyó el operador de elitismo para asegurar la conservación de los mejores alineamientos.

Además, se implementó un mecanismo de autoadaptación para elegir la probabilidad con que se eligen los cinco suboperadores de mutación para su uso.

4.2.1. Representación de los Individuos

En GGGA, los individuos de la población son representaciones de posibles alineamientos. Un alineamiento de múltiples secuencias genéticas puede ser representado mediante una matriz, donde cada fila corresponde a una secuencia o taxa y cada columna a una posición en el alineamiento. Como la longitud de las secuencias en un alineamiento puede no ser la misma, al final de cada secuencia se insertan codificaciones de gap "$-$" hasta que todas alcanzan la longuitud de la secuencia más larga. Una matriz correspondiente a un alineamiento de ejemplo se muestra en la Tabla 4.1.

A partir de la matriz descrita se obtiene la representación de un alineamiento para el algoritmo genético. Como al alinear múltiples secuencias no se modifican las bases que conforman las secuencias, solamente se insertan gaps ("$-$") para alinear las posiciones, son la posición y número de estas codificaciones de gaps las que definen un alineamiento y a partir de ellas se puede generar una representación.

[1] Algoritmo Genético Guidado por GLOCSA

Tabla 4.1: Ejemplo de un alineamiento

secuencia-#	0	1	2	3	4	5	6	7	8
0	A	A	-	-	-	-	A	A	-
1	A	A	-	-	A	-	A	A	A
2	-	-	A	A	A	A	-	-	-
3	A	A	A	-	-	-	A	A	-

Tabla 4.2: Representacion del alineamiento de ejemplo en la Tabla 4.1

secuencia-#	
0	$[2, 4]$
1	$[2, 2]$, $[3, 1]$
2	$[0, 2]$
3	$[3, 3]$

Para simplificar aún más las cosas, las codificaciones individuales de "−" contiguas se pueden agrupar en bloques, los cuales para indicarlos solo haría falta dar su ubicación y tamaño. Una forma eficaz de señalar su ubicación es nombrando a que base en particular preceden. Es decir, si tomamos la secuencia sin ningún "−" (sin alinear, podría decirse), se puede indexar cada base respecto a su posición en esta secuencia no alineada, de manera que cada base se puede indicar por un entero consecutivo. Así, usando el número consecutivo de la base a la que preceden es posible señalar la ubicación de cada bloque de "−" contiguos, que aunado con su tamaño nos dan una descripción inequívoca de cada uno de ellos. Es de notarse que de esta manera no es posible representar los gaps finales (el bloque de "−" contiguos al final de una secuencia), pero representarlos es completamente innecesario, ya que se sabe que estos siempre existirán para completar las secuencias de manera que todas tengan la longuitud de la más larga y formar perfectamente la matriz descrita anteriormente.

Por ejemplo, en el alineamiento de la Tabla 4.1, la secuencia 0 tiene cuatro codificaciones individuales de gap contiguas en las posiciones 2, 3, 4 y 5, por lo que se agrupan en un bloque que se indica por $[2, 4]$, donde 2 es la posición donde comienza el bloque y 4 la longitud del mismo.

De esta manera se pueden codificar todos las ocurrencias de "−" en cada secuencia y tener una representación compacta del alineamiento.

Un ejemplo de esta representación para el alineamiento de ejemplo de la Tabla 4.1 se puede observar en la Tabla 4.2. En este caso, la secuecia 0 tiene un único gap indicado por $[2, 4]$; la secuencia 1 contiene dos gaps, uno de dos posiciones precediendo la base número 2 indicado por $[2, 2]$ y otro de una posición antes de la base número 3 indicado por $[3, 1]$; la secuencia 2 sólo cuenta con un gap inicial de dos posiciones indicado por $[0, 2]$ (al ser inicial precede a la base 0); y finalmente la secuencia 3 tiene un único gap antes de la base número 3 y tiene un tamaño de tres, se indica por $[3, 3]$.

Así con una lista de todos los gaps (bloques de codificaciones "−" contiguas) para cada secuencia, se puede representar un alineamiento de un conjunto de

secuencias, y esta es la representación utilizada en GGGA.

4.2.2. Operador de Mutación

En el operador de mutación recae la responsabilidad de que el algoritmo explore el espacio de búsqueda. Como es estándar en los algoritmos genéticos, éste se aplica con una probabilidad determinada como parámetro del algoritmo, haciendo posible probabilidades que impliquen más de una mutación por individuo, de hecho la probabilidad de mutación se indica por el número esperado de mutaciones en todo el alineamiento. El operador cuenta a su vez con cinco suboperadores diferentes de los cuales en cada aplicación se elige uno para realizar la mutación. La elección del suboperador a utilizar se hace en base a una probabilidad asignada a cada uno; esta probabilidad es adaptada automáticamente a lo largo de la ejecución del algoritmo. Este proceso se detalla en la sección 4.2.3.

Los cinco subperadores de mutación, que se describen a continuación, son:

- Suboperador de inserción de gap
- Suboperador de eliminación de gap
- Suboperador de aumento de tamaño de gap
- Suboperador de disminución de tamaño de gap
- Suboperador de intercambio de gaps

Estos fueron elegidos por considerarse que con ellos es posible explorar el espacio de búsqueda de los posibles alineamientos de manera relativamente eficiente.

Suboperador de inserción de gap

Al aplicarse este suboperador, se elige al azar una secuencia y se inserta un gap (un bloque de codificaciones "−" contiguas) en ella. La ubicación (base a la que precede) del nuevo gap también es elegida aleatoriamente entre las ubicaciones sin gap. Si se da el caso de que a cada base la precede un gap, se inserta en una ubicación totalmente al azar.

El tamaño del gap se determina de manera aleatoria pero con un sesgo por gaps de pocas codificaciones. Esto debido a que gaps demasiado largos son inusuales, aunque no imposibles. La forma en que se implementa la tendencia por tamaños pequeños es a través de la ecuación 4.1.

$$f_{br}(min, max, x) = (max - min)\left(\frac{e^x - 1}{e^k - 1}\right) + min \tag{4.1}$$

donde max y min son el máximo y el mínimo que delimitan el rango entre los cuales se busca el tamaño, mientras x es un número aleatorio entre 0 y 1 a partir del cual se genera el número aleatorio con tendencia a números cercanos

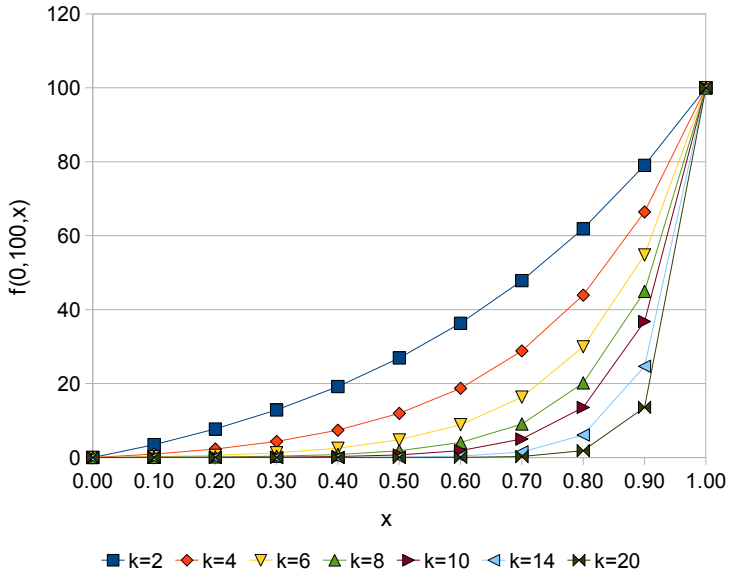


Figura 4.1: Función de sesgo utilizada para determinar el tamaño de un nuevo *gap*.

al límite inferior. La fuerza del sesgo se regula por medio de la variable k. En la Figura 4.1 se expone su efecto en el resultado, se muestra como a partir de la variable de entrada x se va sesgando la salida de la función con base en el valor de k. Mientras k sea más grande, la tendencia hacia números más cercanos al límite inferior será mayor (cuando $k = 1$ no existe sesgo alguno).

Suboperador de eliminación de gap

Una aplicación de este suboperador consiste en elegir una secuencia al azar y eliminar un gap (bloque de codificaciones "−" contiguas) de ella. Si no existe gap alguno en la secuencia seleccionada no se realiza operación alguna.

Suboperador de aumento de tamaño de gap

Cuando se aplica este suboperador, se elige una secuencia al azar, y de esa secuencia se elige un gap (bloque de codificaciones "−" contiguas) existente, y se incrementa su tamaño en una unidad. Si no existe gap alguno en la secuencia no se realiza ninguna operación.

Suboperador de disminución de tamaño de gap

Al aplicarse este suboperador, se elige una secuencia aleatoriamiente, para después de esa secuencia elegir un gap (bloque de codificaciones "−" contiguas)

Tabla 4.3: Registros de ejemplo utilizados para el proceso de adaptación

	suboperador					Puntuación GLOCSA	
	0	1	2	3	4	Puntación Anterior	Puntuación Posterior
				...			
15	0	1	1	0	0	34.9	34.5
16	1	0	1	0	0	34.9	35.1
				...			

existente y decrementar su tamaño en una unidad. Si el tamaño del gap eligido es de una sola codificación, el gap se elimina totalmente. Si no existe gap alguno en la secuencia no se realiza ninguna operación.

Suboperador de intercambio de gaps

Aplicar este suboperador implica elegir una secuencia al azar, y de esta un gap (bloque de codificaciones "−" contiguas) existente. Después se elige otra ubicación aleatoriamente en la misma secuencia, si en esa ubicación no existe gap, se inserta ahí un gap del tamaño del seleccionado previamente, y se elimina el seleccionado primero. Si ya existe ahí otro gap, se intercambian los tamaños de los gaps entre ellos.

4.2.3. Adaptación de los Suboperadores de Mutación

La probabilidad de aplicar cada uno de los suboperadores de mutación se adapta dinámicamente a través de las generaciones. Va cambiando de acuerdo a su efecto en la puntuación de GLOCSA que causa en los alineamientos que representan los individuos de la población, buscando que los suboperadores con un efecto más benéfico tengan una probabilidad mayor de aplicarse.

El proceso de identificar los suboperadores mas benéficos y adaptar su probabilidad en consecuencia, se realiza al final de cada generación y el proceso se describe a continuación.

Para la primera generación del algoritmo genético los cinco suboperadores tienen la misma probabilidad, cada uno con probabilidad de 0.20 de ser utilizado. Cada vez que el operador de mutación se aplica, en un *registro* se guardan los valores de GLOCSA correspondientes al alineamiento que el individuo representa antes y después de aplicar el operador. Ésto con el objetivo de registrar el efecto de la mutación en el alineamiento. Así mismo también se registra qué suboperadores fueron utilizados, a lo que se hace referencia como la cuenta de uso de suboperador. Un ejemplo de este tipo de registro se muestra en la Tabla 4.3.

Después de generar la nueva población, al final de una generación, comienza el proceso de adaptar las probabilidades de los suboperadores. Primero se calcula una variable, para cada *registro*, que por fines explicativos se llamará *diferencia atribuida por suboperador*. Así, para cada registro se divide el número de veces que se usó cada suboperador entre el número total de veces que se usaron todos

los suboperadores (total de mutaciones), y después se multiplica por la diferencia entre las puntuaciones de GLOCSA posterior y anterior. Esto se muestra en la ecuación 4.2, donde dSO_s^r es la *diferencia atribuida por suboperador* para el suboperador $s \in S$ (S es el conjunto de suboperadores) en el registro $r \in R$ (R es el conjunto de registros), $sOUC_s^r$ es la cuenta de uso del suboperador s en el registro r, tM^r el total de mutaciones llevadas a cabo en el registro r, ($tM^r = \sum sOUC_s^r$), aS^r y bS^r son las puntaciones de GLOCSA posterior y anterior a la aplicación del operador de mutación en el registro r, respectivamente.

$$dSO_s^r = \left(\frac{sOUC_s^r}{tM^r}\right)(aS^r - bS^r) \tag{4.2}$$

$$\forall s, s \in S$$

$$\forall r, r \in R$$

Entonces, la *diferencia atribuida por suboperador* de cada registro se suma en la *diferencia total atribuida por suboperador* ($tDSO_s$, ecuación 4.3)

$$tDSO_s = \sum_r dSO_s^r \tag{4.3}$$

$$\forall s, s \in S$$

Los valores $tDSO_s$ son normalizados, dividiéndolos entre el valor absoluto más grande de ellos,

$$tDSO_s = \frac{tDSO_s}{max\left(\{|tDSO_s| \quad \forall s\}\right)} \tag{4.4}$$

Después, a la probabilidad de cada suboperador p_s se le suma $p_s \cdot SCh \cdot tDSO_s$,

$$p_s(t+1) = p_s(t) + (p_s(t) \cdot SCh \cdot tDSO_s(t))$$

donde SCh es una constante que especifica cuanto se modifica la probabilidad de mutación de acuerdo a $tDSO_s$, en este caso se utilizó $SCh = 0.10$.

Y finalmente los valores de p_s son escalados para hacer que la suma de todas las probabilidades sea 1.

$$p_s = \frac{p_s}{\sum_S p_s}$$

4.2.4. Inicialización de la población

Para inicializar la población de la generación 0, se usa un alineamiento como punto de partida. Este alineamiento puede bien ser el resultado de un programa de alineamiento existente, como MUSCLE [16], KALIGN [31], CLUSTAL [39]. A partir de el alineamiento de entrada, la población inicial se genera a partir de mutaciones de este, y un individuo siendo el alineamiento mismo (en una especie de elitismo inicial). Para crear las mutaciones se usa el operador de mutación descrito en la sección 4.2.2, sin realizar el proceso de adaptación de las probabilidades de los suboperadores, y teniendo para cada suboperador la misma probabilidad de aplicación.

Capítulo 5

Pruebas y Resultados de GGGA

Teniendo el criterio de evaluación para los alineamientos de secuencias, GLOCSA y el algoritmo génetico adaptado para el problema usando a GLOCSA como función objetivo, se probó su capacidad para mejorar los alineamientos de secuencias generados por otro programa de alineamiento, MUSCLE, que cuenta con relativa popularidad, el cual es específicamente un programa de alineamiento progresivo [16].

5.1. Conjuntos de Pruebas

En esta etapa se propusieron tres conjuntos de secuencias para probar la capacidad de GGGA de mejorar los resultados que entrega MUSCLE. En la Tabla 5.1 se listan éstos y cierta información sobre ellos, como el número de secuencias, el número máximo de bases en una secuencia y el número total de bases en todo el alineamiento.

La decisión del número de conjuntos de prueba y su elección se hizo con base en la posibilidad de realizar suficientes pruebas, variando la probabilidad de mutación del algoritmo, para dar también una idea del impacto de este operador.

El conjunto de secuencias *exmpl17* es un subconjunto de *exmpl19*, donde las dos secuencias más pequeñas (considerablemente más pequeñas que el resto) fueron eliminadas, reduciendo entonces la complejidad del alineamiento.

5.2. Parámetros de GGGA

Cada conjunto de secuencias fue alineado en primera instancia usando MUSCLE [16], en su versión 3.6, con los parámetros por default. El alineamiento

Tabla 5.1: Conjunto de Pruebas

	# de secuencias	# máximo de bases	# total de bases
exmpl19	19	649	10908
exmpl17	17	649	10149
exmpl29	29	245	6150

resultante se usó como punto de partida para la generación de la población inicial.

El algoritmo genético para todos los experimentos se ejecutó durante 1000 generaciones, con una población de 100 individuos. Se utilizó *elitismo*, pasando directamente a la siguiente generación los 5 mejores individuos de la población actual. La *selección* se llevó a cabo usando torneo de 5 individuos.

La función de sesgo utilizada para determinar el tamaño de los nuevos gaps a insertar por el operador de mutación, se usó con $k = 10$ para definir la fuerza de la tendencia hacia gaps de menor tamaño, con un mínimo de 1 posición y un máximo definido por el número de bases de la secuencia más larga. Para la inicialización de la población se utilizan los mismos parámetros.

La probabilidad de mutación se probó en el rango que implica de $[0.1, 3]$ mutaciones esperadas por alineamiento, con incrementos de 0.1 de las mismas unidades. Para cada combinación de la probabilidad de mutación y los parámetros antes definidos, se realizarón 30 experimientos.

La función de evaluación de la calidad de los alineamientos, GLOCSA, usada como función objetivo se aplicó con los parámetros por default, definidos en la Tabla 3.1.

5.3. Resultados

En las Figuras 5.1, 5.2 y 5.3 se muestran los resultados de los experimentos antes descritos. En los 30 experimentos que se realizaron con diferentes valores de probabilidad de mutación en el rango de $[0.1, 3]$, se registró la puntación GLOCSA del mejor alineamiento obtenido al final de las 1000 generaciones en las que se ejecutó el algoritmo. Se muestra el valor mínimo, el promedio y el máximo de estos resultados para los 30 experimentos.

En ellas se puede observar que el Algoritmo Genético Guiado por GLOCSA siempre logró mejorar el alineamiento que se había realizado previemente usando MUSCLE. También se observa que la mejora que se obtiene depende en gran medida de la probabilidad de mutación con que se ejecute el algoritmo. Cuando la probabilidad de mutación tiende a 0 mutaciones por alineamiento, y por consiguiente la exploración del espacio de búsqueda se limita, las mejoras al alineamiento en cuestión son menores. Mientras que cuando la mutación se da con mayor probabilidad, acercándose a 3 mutaciones por alineamiento y mas allá, las mejoras también disminuyen, esta vez por que los individuos se

Figura 5.1:

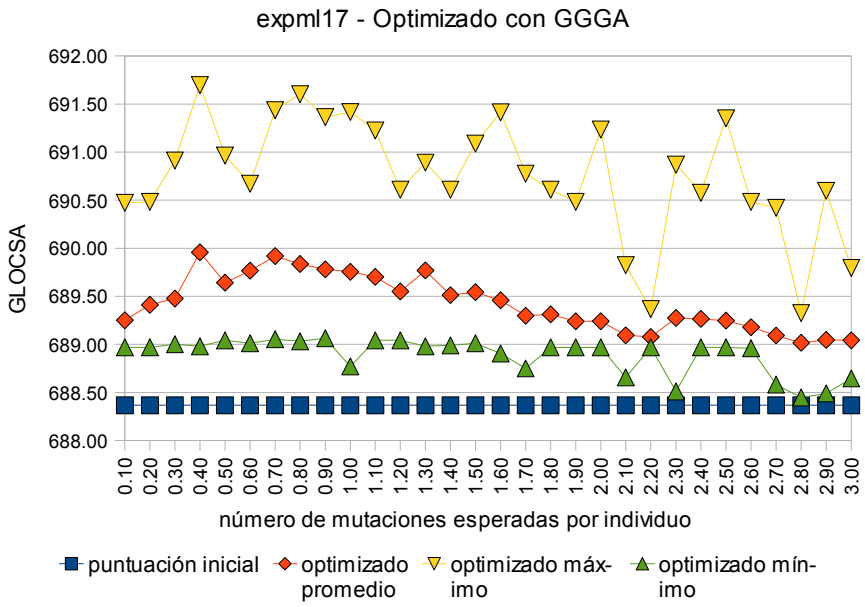


Figura 5.2:

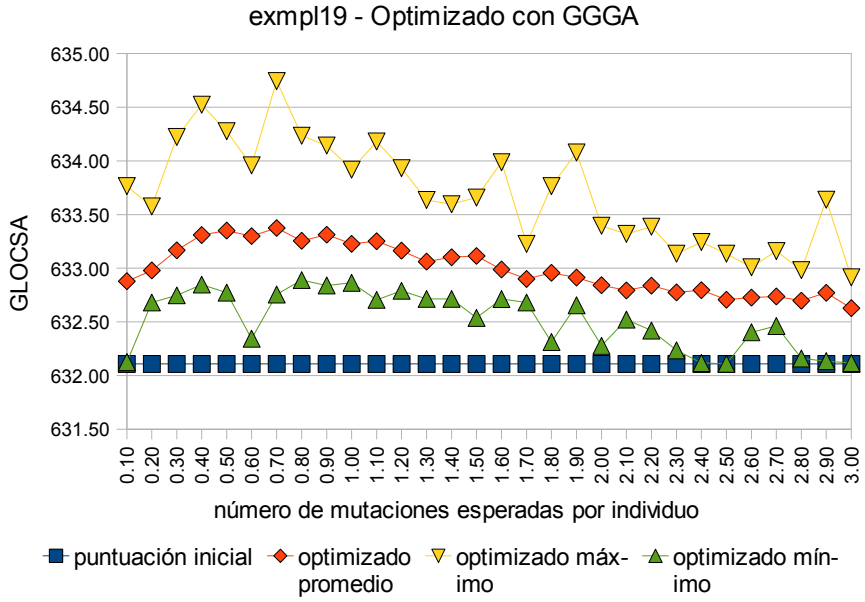


Figura 5.3:

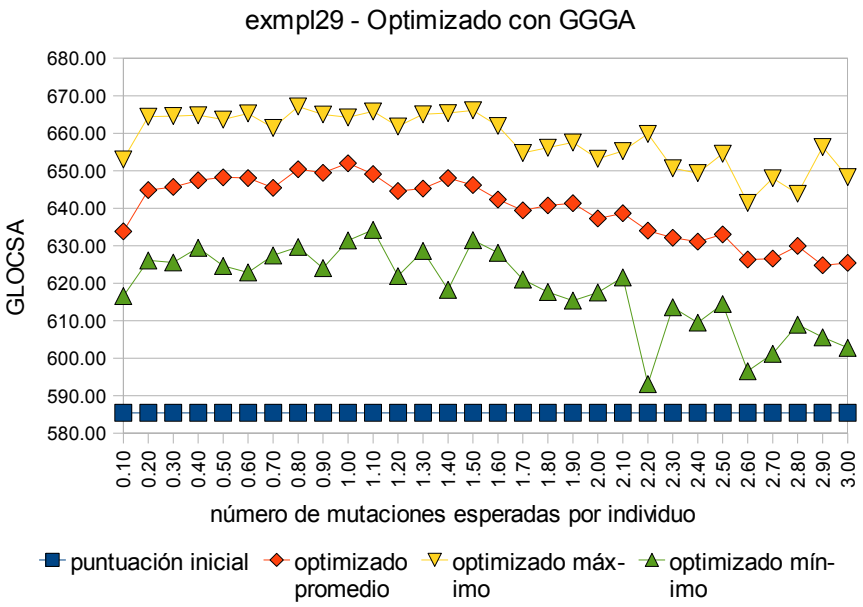


Tabla 5.2: Tiempos de experimentos

prueba	tiempo (segundos)
exmpl19	1188.74
exmpl17	1153.44
exmpl29	643.37

están mutando demasiado. En cambio, en el rango de $[0.5, 1.0]$ mutaciones por alineamiento es donde se obtienen los mejores resultados, hecho que se puede observar claramente en el promedio de los resultados del proceso de optimización.

En la Tabla 5.2 se muestran los tiempos promedio en lo que se ejecutaron los experimentos para cada conjunto de secuencias. Todos los experimentos se realizaron utilizando una computadora personal ordinaria y reciente.

Capítulo 6

Conclusiones

Encontrar una métrica para evaluar alineamientos de múltiples secuencias genéticas, que sea independiente de librerías, o comparaciones entre otros alineamientos, es un problema importante y que vale la pena abordar. Se han propuesto soluciones, pero con GLOCSA se busca tener una métrica simple que evalue el alineamiento como un todo, y que incorpore conceptos de parsimonia, criterios que se consideran importantes al calificar alineamientos de secuencias.

Al realizar los experimentos, se hace evidente que cuando el espacio de posibles soluciones es de un tamaño considerable (por supuesto relativo al poder de cómputo disponible en determinado momento), la necesidad de incorporar información específica del domino del problema se hace más imperiosa. A pesar de que el operador de mutación se creó específicamente para el problema y que se diseñó una representación que reduce el espacio de búsqueda dejando solo la información necesaria para reconstruir el alineamiento, el algoritmo genético requiere mucho más tiempo para hacer optimizaciones, comparado con el que toma alinear el conjunto de secuencias con algoritmos heurísticos de alineamiento progresivo como el de MUSCLE.

Por otro lado, un algoritmo genético al ser una herramienta de optimización general nos permite utilizar el criterio GLOCSA para guiar el proceso de búsqueda. Además de que utilizando como punto de partida alineamientos previamente realizados por otros programas, nos da la posibilidad de utilizarlo como herramienta de refinamiento de alineamientos usando los criterios que GLOCSA representa.

Si bien los algoritmos genéticos (y en general el cómputo evolutivo) son una herramienta muy poderosa, no se debe perder de vista que son una técnica de optimización general, que en muchas ocasiones es superada por algoritmos específicos a los problemas que se están tratando, lo que guarda estrecha relación con el teorema del *no free lunch* [44].

Específicamente sobre la implementación del algoritmo genético guiado por GLOCSA, en los resultados se puede observar claramente lo importante de mantener el equilibrio entre exploración y explotación, ya que los mejores resultados se obtienen con valores intermedios de la probabilidad de mutación, ya que con

poca mutación no se tiene suficiente exploración y con probabilidades altas de mutación la conservación no es suficiente. Este concepto, se considera de fundamental importancia para el desempeño de los algoritmos genéticos.

6.1. Trabajo Futuro

El criterio de evaluación GLOCSA, fue validado empíricamente, calibrando los pesos e incluyendo los criterios individuales, lo cual deja abierta la línea de una validación mucho más exhaustiva y rigurosa. Es posible hacer uso de *benchmarks* [43] [4] existentes para evaluar el resultado de programas de alineamiento, para indirectamente probar la correlación de los valores de GLOCSA y alineamientos considerados como mejores resultados por estos conjuntos de prueba. Aunque para ésto es necesario adaptar GLOCSA a evaluar secuencias de proteínas.

La incorporación de un enfoque de optimización multiobjetivo podría resultar benéfica al poder evaluar los criterios individuales de GLOCSA sin tener que definir los pesos necesarios para su consideración como un solo valor.

Así mismo, es evidente que es posible ajustar el algoritmo genético para explorar el espacio de búsqueda de una manera mucho más eficiente, incorporando nuevos operadores que resulten en una mejor exploración, seguir la experimentación e investigación sobre la autoadaptación de los operadores de mutación, así como la utilización de esquemas como el propuesto por el RankGA [9] para la asignación de la probabilidad de mutación. También, utilizar técnicas de multipoblación con miras a distribuir la carga computacional en más hardware, redituaría en disminuir el tiempo requerido por el algoritmo.

Todo lo anterior puede ser considerado como una guía para trabajo futuro.

Apéndice A

Herramientas basadas en GLOCSA

A.1. GLOCSAWEB: Implementación de un Evaluador en línea

El criterio descrito aquí se encuentra implementado actualmente en una aplicación web, llamada `glocsaweb`, a la cual se le alimenta un alineamiento en formato FASTA [38], como se puede observar en la Figura A.1 y regresa por medio de una página web su evaluación, e.g en la Figura A.2. Ahí se muestran ciertos datos informativos sobre el alineamiento, los valores para cada criterio que forma GLOCSA, y GLOCSA mismo, además del número de sinapomorfías potenciales.

A.1.1. Sinapomorfías potenciales

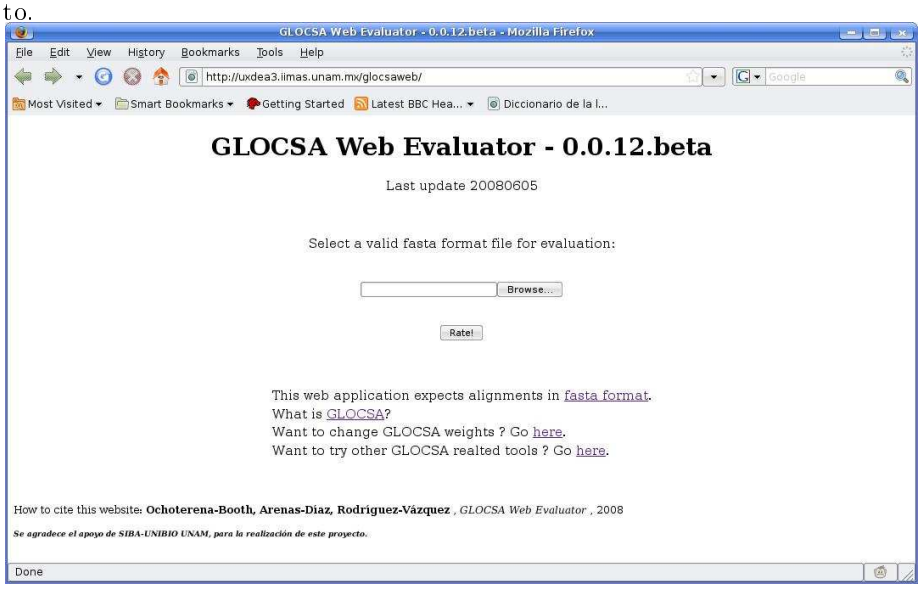
La cuenta de las sinapomorfías potenciales se realiza en dos partes, una para las implicadas por alguna *substitución* de base y otra para las implicadas por algún *indel*.

El proceso de contar las sinapomorfías ligadas a las *substituciones* se realiza columna por columna, tomando en cuenta el número de bases diferentes en cada columna y restándole una unidad, para finalmente sumar las substituciones implicadas de todas las columnas. En el caso de que estén presentes polimorfismos[1] en alguna columna, se consideran las cuentas pesadas que se utilzan en el cálculo de la homogeneidad de columna (ver sección 3.1) para tomar como bases existentes las que tienen un mayor valor en la cuenta pesada y que satisfagan todos los polimorfismos encontrados en la columna en cuestión.

La cuenta de las sinapomorfías correspondientes a *indels* se realiza usando

[1] codificaciones que indican que puede ser una entre un conjunto de bases, generalmente por imprecisiones en la lectura de la secuencia

Figura A.1: Página princiapl de `glocsaweb`, entrada del archivo del alineamiento.



la *codificación de gaps simple*[2] [41], tomando cada *caracter* de la codificación como una sinapomorfía.

A.1.2. Gráficas y matrices

Es posible también observar en detalle la matriz del alineamiento (Figura A.3), y una matriz con una representación conocida como *codificación de gaps simple* [41] (Figura A.4)

Además se pueden observar algunas gráficas informativas sobre la composición del alineamiento. Como qué proporción de columnas tienen una, dos, tres o cuatro bases diferentes y/o gaps (Figura A.5); el tamaño y frecuencia de los bloques de gaps en el alineamiento (Figura A.6) y la puntación del criterio de homogeneidad de columna a lo largo del alineamiento (Figura A.7).

La aplicación se encuentra disponible en la siguiente dirección web:
`http://uxdea3.iimas.unam.mx:/glocsaweb/`

A.2. AT: Herramientas de Alineamiento

También se implementaron un conjunto de programas para ayudar en las tareas de alineamiento, agrupadas en una sola aplicación llamada genéricamente *AlignmentTools* (Figura A.8). Estas herramientas son tres programas: *Glocser*

[2] simple gap coding

Figura A.2: Resultado de la evaluación en `glocsaweb`.

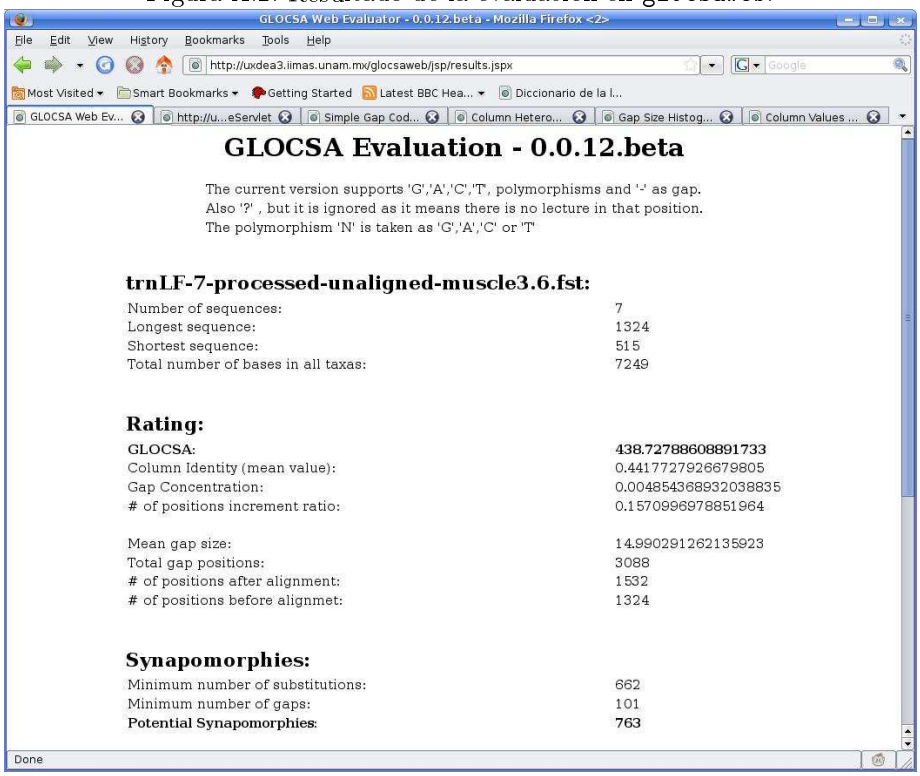


Figura A.3: Detalle de la matriz del alineamiento.

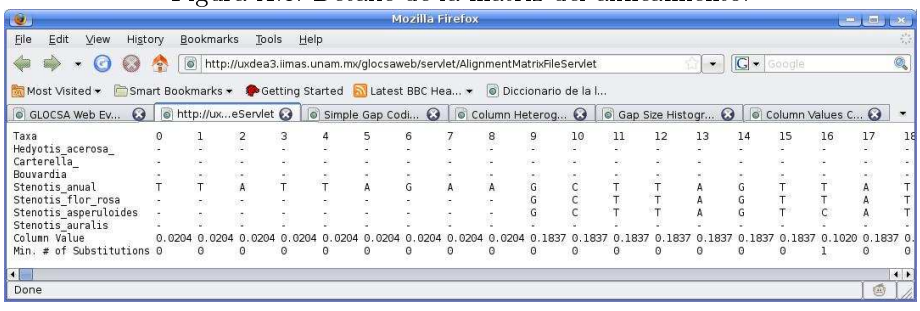


Figura A.4: Matriz con la codificación simple de gaps.

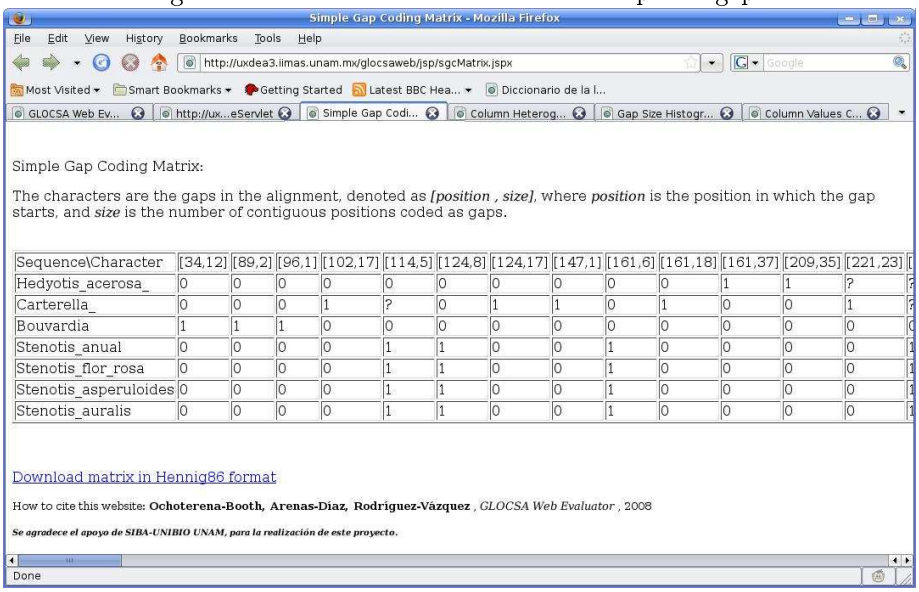


Figura A.5: Gráfica informativa sobre la composición de las columnas.

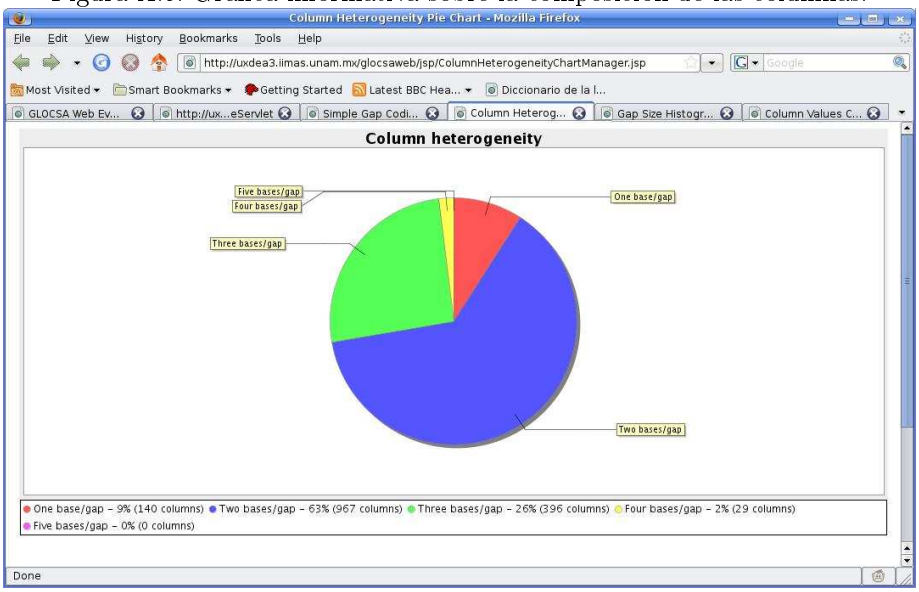


Figura A.6: Gráfica informativa sobre el tamaño y frecuencia de los bloques de gaps.

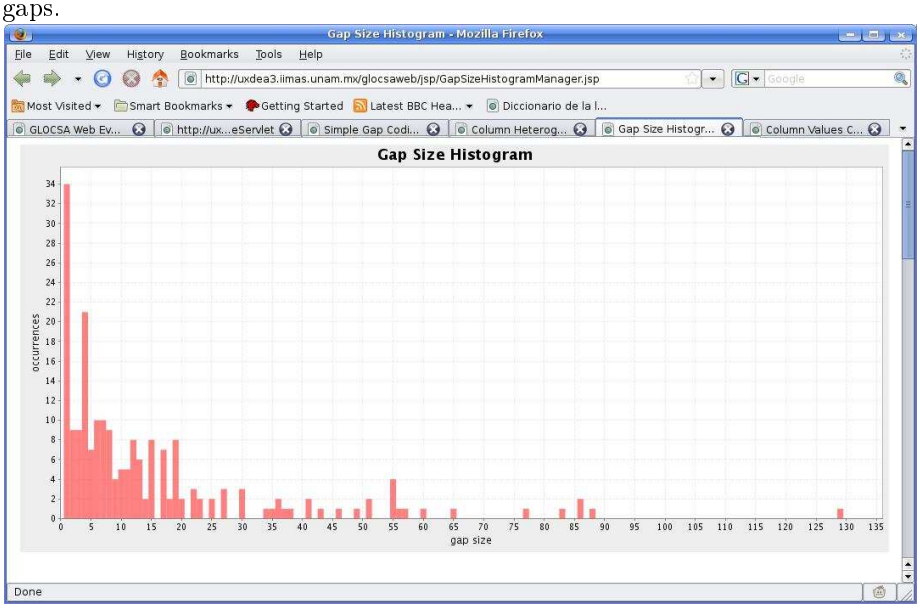


Figura A.7: Gráfica informativa sobre el valor de homogeneidad de columna.

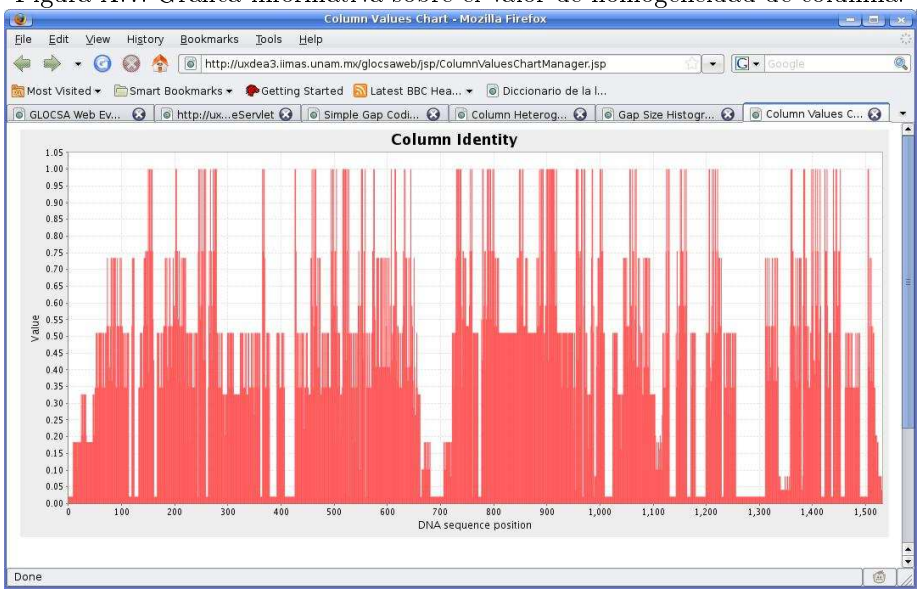
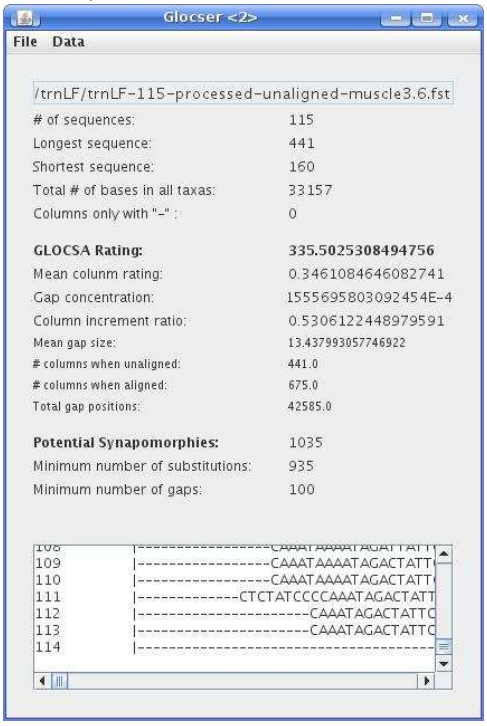


Figura A.8: Herramientas de Alineamiento.

Figura A.9: Glocser, evaluador de alineamientos usando GLOCSA.



que evalúa un alineamiento usando GLOCSA (Figura A.9), una herramienta de comparación entre dos alineamientos sobre el mismo conjunto de secuencias (Figura A.10) y una herramienta más para convertir archivos de alineamientos en el formato Hennig86 al formato FASTA (Figura A.11).

A.3. Comparación gráfica de alineamientos

Una forma práctica para comparar alineamientos entre si, es de manera gráfica, usando una gráfica de red con los datos de la evaluación de cada alineamiento de acuerdo a los criterios que se están tomando en cuenta.

Antes de alimentar la gráfica se realiza un escalamiento de los datos. Para los criterios que se buscan maximizar, el valor se divide entre el valor para ese

Figura A.10: Herramienta de comparación entre alineamientos.

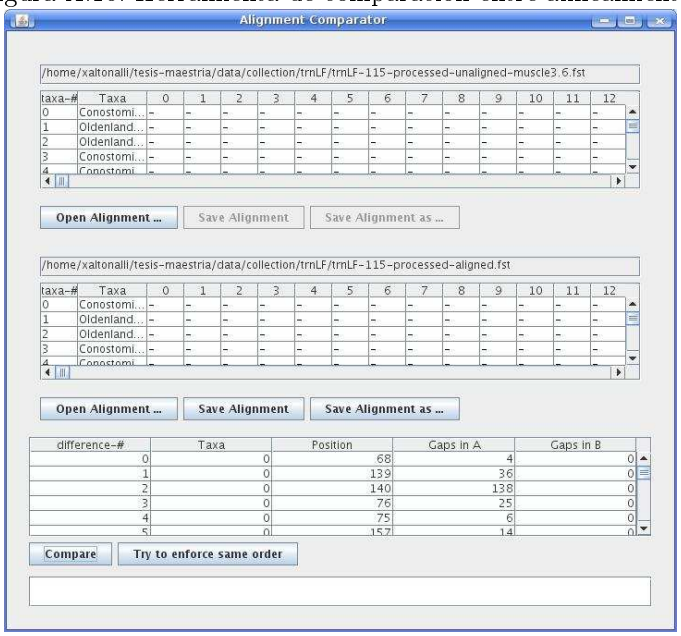


Figura A.11: Convertidor de Hennig86 a FASTA.

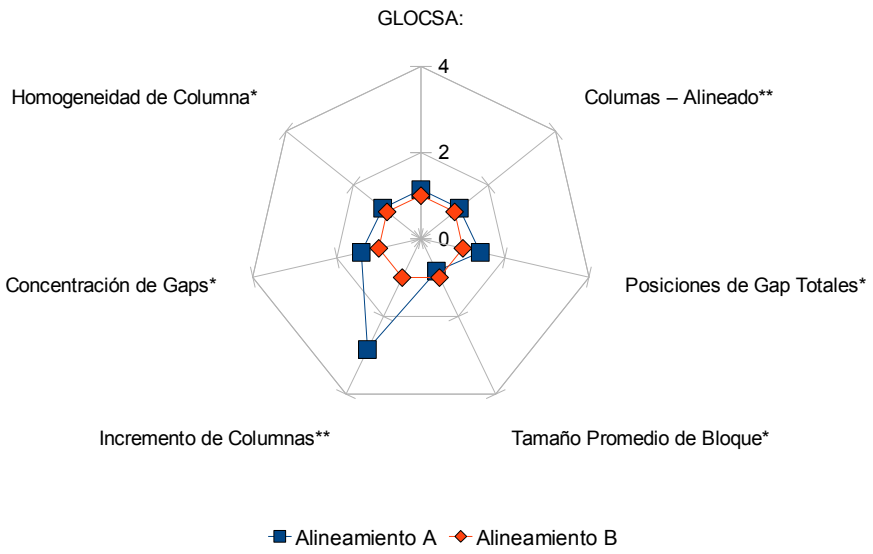


Figura A.12: Comparación gráfica de dos alineamientos. [*] Cantidades divididas entre el valor para el alineamiento A. [**] Recíproco de las cantidades divididas entre el valor para el alineamiento A.

criterio de un alineamiento específico que es considerado como referencia. Y para los criterios que se buscan minimizar, se utiliza el resultado de dividir el valor del criterio en el alineamiento de referencia entre el valor del criterio del alineamiento (i.e. el recíproco de lo que se utiliza para los criterios que se buscan maximizar). La diferencia entre el proceso de escalado para los criterios a maximizar y a minimizar se debe a que de esta manera, en la gráfica de red, un mejor valor para cualquier criterio siempre estará por fuera. Es de notar también que con este procedimiento de escalamiento, los valores del alinemiento de referencia siempre serán 1, sirviendo efectivamente como una línea de referencia en la gráfica de red.

Por ejemplo, si se van a comparar dos alineamientos, alineamiento A y alineamiento B, y el alineamiento A será considerado como referencia, los valores para la homogeneidad de columna del alineamiento B, un criterio que se busca maximizar, se dividirán entre el valor de homogeneidad de columna del alineamiento A. Mientras que para el criterio de incremento de columnas se divide el valor del alineamiento A entre el valor del alineamiento B. El resultado se ejemplifica en la figura A.12 para GLOCSA, todos los criterios que lo conforman y las subunidades con los que se calculan.

Bibliografía

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[2] S. F. Altschul, T. L. Madden, A. A. Schäfer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[3] S. Anderson. Shotgun dna sequencing using cloned dnase i-generated fragments. *Nucleic Aids Research*, 9(13), 1981.

[4] A. Bahr, J.D. Thompson, J.C. Thierry, and O. Poch. Balibase(benchmark alignment database): enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acids Reseach*, 29:323–326, 2001.

[5] Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and David L. Wheeler. Genbank. *Nucleic Acids Reseach*, 34:D16–D20, 2006.

[6] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer Jr., M.D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank: A computer-based archival file for macromolecular structures. *Journal of Molecular Biology*, 112:535–542, 1977.

[7] C. Branden and J. Tooze. *Introduction to Protein Structure.* Garland Publishing Inc., New York, 2 edition, 1998.

[8] L. Cai, D. Juedes, and E. Liaknovitch. Evolutionary computation techniques for multiple sequence alignment. In *Proceedings of the IEEE Congress on Evolutionary Computation 2000*, 2000.

[9] Jorge Cervantes and Christopher Rhodes Stephens. Rank based variation operators for genetic algorithms. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation.* ACM, New York, NY, USA, 2008.

[10] Lance D. Chambers, editor. *The Practical Handbook of Genetic Algorithms: Applications.* Chapman and Hall / CRC, 2000.

[11] K. Chellapilla and G.B. Fogel. Multiple sequence alignment using evolutionary programming. In *Proceedings of the IEEE Congress on Evolutionary Computation 2000*, 1999.

[12] Francis Harry Compton Crick. On protein synthesis. In *Symp. Soc. Exp. Biol. XII*, pages 139–163, 1958.

[13] Francis Harry Compton Crick. Central dogma of molecular biology. *Nature*, 112:561–563, 1977.

[14] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5(3):345–352, 1978.

[15] A. L. Delcher, S. Kasti, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg. Alignment of whole genomes. *Nucleic Acids Research*, 27:2369–2376, 1999.

[16] Robert C. Edgar. Muscle: multiple sequence alignment with high accurracy and high throughput. *Nucleic Acids Reseach*, 32(5):1792–1797, 2004.

[17] D.-F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25:351–360, 1987.

[18] David B. Fogel, editor. *Evolutionary Computation: The Fossil Record*. IEEE Press, New York, 1998.

[19] Gary B. Fogel and David W. Corne, editors. *Evolutionary Computation in Bioinformatics*. Morgan Kaufman, 2003.

[20] Gary B. Fogel and David W. Corne, editors. *Evolutionary Computation in Bioinformatics*, chapter 5: Using Genetic Algorithms for Pairwise and Multiple Sequence Alignments. Morgan Kaufman, 2003.

[21] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, 1966.

[22] D.E. Glodberg. *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, New York, 1989.

[23] Bernhard Haubold and Thomas Wiehe. *Introduction to Computational Biology: An evolutionary approach*. Brikhäuser Basel, 2007.

[24] Steven Henikoff and Jorja G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89(22):10915–10919, 1992.

[25] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[26] M. Ishikawa, T. Toya, M. Hoshida, Ogiwara A. Nitta, K., and M Kanehisa. Multiple sequence alignment by parallel simulated annealing. In *CABIOS*, number 9, pages 267–273, 1993.

[27] M. Ishikawa, T. Toya, and Y. Tokoti. Parallel iterative aligner with genetic algorithm. In *Artificial Ingelligence and Genome Workshop, 13th International Conference on Artificial Intelligence*, 1993.

[28] Charles Karr and L. Michael Freeman, editors. *Industrial Applications of Genetic Algorithms*. CRC, 1998.

[29] H Kitano. *Foundations of System Biology*. MIT Press, Cambridge Massachussetts, 2001.

[30] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[31] T. Lassmann and Erik L.L. Sonnhammer. Kalign - an accurate and fast multiple sequence alignment algorithm. *Nucleic Acids Research*, 6:298, 2005.

[32] B. Lewin. *Genes VII*. Oxford University Press, New York, 2001.

[33] C.R. Marshall and J.W. Schopf. *Evolution and the Molecular Revolution*. Jones and Bartlett Publisher International, London, 1996.

[34] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[35] Cedric Notredame and D.G. Higgins. Saga: sequence alignment by genetic algorithm. *Nucleic Acids Research*, 1996.

[36] Cedric Notredame, L. Hols, and D.G. Higgins. Coffee: an objective function for multiple sequence alignments. *Bioinformatics*, 14:407–422, 1998.

[37] Cedric Notredame, E.A. O'Brien, and D.G. Higgins. Raga: Rna sequence alignment by genetic algorithm. *Nucleic Acids Research*, 25:4570–4580, 1997.

[38] W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 85(8):2444–2448, 1988.

[39] Chenna R., Sugawara H., Koike T., Lopez R., Gibson T.J., Higgins D.G., and Thompson J.D. Multiple sequence alignment with the clustal series of programs. *Nucleic Acids Research*, 31:3497–3500, 2003.

[40] Ingo Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzboog, 1973.

[41] Mark P. Simmons and Helga Ochoterena. Gaps as characters in sequence-based phylogenetic analyses. *Systematic Biology*, 49(2), 2000.

[42] T.F. Smith and M.S Waterman. Comparision of biosequences. *Adv. Appl. Math.*, 2:483–489, 1981.

[43] J.D. Thompson, F. Plewniak, and O. Poch. Balibase: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15:87–88, 1999.

[44] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 67(1), 1997.