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

DETECTOR DE CRISPR EN GENOMA DE BACTERIAS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL EN COMPUTACIÓN

MATILDE RIVAS LAGOS

PROFESOR GUÍA:
GONZALO NAVARRO BADINO

MIEMBROS DE LA COMISIÓN:
FEDERICO OLMEDO BERÓN
JOSÉ M. PIQUER GARDNER

Este trabajo ha sido parcialmente financiado por Proyecto Fondecyt 1-170048

SANTIAGO DE CHILE
2020

Resumen

Las repeticiones palindrómicas cortas agrupadas y regularmente interespaciadas (CRISPR) son secuencias de ADN encontradas en el código genético de microbios. Estas secuencias forman parte del sistema inmunológico de las bacterias, guardando fragmentos de ADN de virus invasores que permiten al organismo defenderse ante futuros ataques. Una zona CRISPR se compone de una serie de repeticiones de un patrón, separadas por secuencias llamadas *spacers*.

La detección de CRISPR en genomas de bacterias extraídas directamente del ambiente, i.e. metagenomas, permite conocer las interacciones entre microbios y el lugar en el que habitan. En particular en el Desierto de Atacama estudiar estas interacciones es de interés dado a las características extremas que presenta este habitat.

En este Trabajo de Título se busca determinar si es posible mejorar el rendimiento de herramientas detectoras de CRISPR *de novo* mediante el uso de estructuras de datos compactas. Para ello se propone diseñar e implementar una nueva estrategia de detección de CRISPR y compararla con algunas existentes. Además se desea utilizar estas herramientas para encontrar CRISPR en genomas obtenidos del Desierto de Atacama.

En específico, la solución implementada se compone de una selección de candidatos a CRISPR utilizando un árbol de sufijos compacto para detectar patrones repetidos en en genoma y una verificación y posterior concatenación de estos usando un árbol wavelet.

Se validó el algoritmo propuesto realizando pruebas de calidad, contabilizando la cantidad de CRISPR conocidos que fue capaz de encontrar en genomas ya estudiados. Se obtuvo una precisión parcial promedio de 65 % y un recall parcial promedio de 95 %. Se comparó el rendimiento de la solución con herramientas existentes en cuanto a uso de recursos. La estrategia implementada utiliza menos memoria durante su ejecución, pero presenta un incremento muy grande con respecto al tiempo de ejecución en comparación con los otros instrumentos.

Utilizando el algoritmo implementado y herramientas existentes se logró encontrar secuencias CRISPR en dos genomas ensamblados a partir de un metagenoma obtenido del Desierto de Atacama. La solución entregó resultados completos en relación a los otros instrumentos, pero también reportó secuencias que estos no consideraron como CRISPR.

Este trabajo de título da un primer acercamiento a la factibilidad de diseñar una herramienta detectora de CRISPR *de novo* que haga uso de estructuras de datos compactas. Se logró implementar una herramienta que garantiza la detección y reporte de repeticiones interespaciadas correspondientes a CRISPR. El trabajo realizado indica que es posible usar estas estructuras para detectar CRISPR en genomas y se proponen ideas para mejorar el rendimiento de la solución propuesta.

Agradecimientos

Muchas gracias a todos mis frens del DCC, a mis amigas incondicionales, a Cristóbal por ser mi debugging duck y especialmente al equipo del Laboratorio de Complejidad Microbiana y Ecología Funcional de la Universidad de Antofagasta por acogerme, responder todas mis dudas y facilitar los datos necesarios para este trabajo.

Gracias también a mi familia por su apoyo y cariño a la distancia.

Tabla de Contenido

Índice de Ilustraciones

Índice de Pseudocódigo

Índice de Tablas

Capítulo 1

Introducción

1.1. Antecedentes

El material genético de todo ser vivo, la información e instrucciones de su desarrollo y funcionamiento, se guarda en su ADN en la forma de largas secuencias de nucleótidos. Estas moléculas difieren únicamente en su base nitrogenada - Adenina(A), Timina(T), Guanina(G) y Citosina(C) - y es el orden de esta secuencia lo que define la genética de un organismo. Es posible extraer el ADN de un organismo y secuenciar su genoma, esto es determinar nucleótido a nucleótido la secuencia de su material genético. La secuenciación del material genético suele hacerse con organismos cultivados en un laboratorio, o *in vitro*. La metagenómica es el estudio del material genético de muestras obtenidas del medio ambiente, es decir, se secuencia el ADN de organismos extraídos de su ambiente, permitiendo estudiar en profundidad la biodiversidad microbiana.

Las repeticiones palindrómicas cortas agrupadas y regularmente interespaciadas (CRISPR por su nombre en inglés) son secuencias de ADN encontradas en el genoma de organismos procariontes como bacterias y arqueas [8]. Las secuencias CRISPR junto con unas proteínas llamadas *cas* actúan como sistema de defensa adaptativo; contienen material genético de virus que han atacado al organismo, permitiendo detectar futuras infecciones y defenderse efectivamente de ellas [18].

Estas zonas de ADN se caracterizan por ser una serie de repeticiones de una misma secuencia de nucleótidos separadas por *spacers* únicos de cierto largo. Estos últimos corresponden al código genético inyectado por la bacteria. A las repeticiones las denominaremos DR, por su nombre en inglés *direct repeats*. El CRISPR se encuentra acompañado de genes *cas* y es precedido por una secuencia de ADN llamada "líder". La Figura 1.1 muestra un esquema de una zona (*locus*) de CRISPR.

Este sistema inmune puede ser usado como herramienta de edición genética en cualquier célula[9], cortando y pegando material genético en el ADN que se desee. Esto tiene variadas aplicaciones en medicina, agricultura y farmacología.

Figura 1.1: Esquema de CRISPR acompañado por genes *cas*. Los DR son separados por distintos *spacers*

1.2. Motivación

La detección *de novo* de CRISPR corresponde a encontrar secuencias CRISPR no documentadas, es decir, desconocidas. Se basa en hallar patrones repetidos de cierto largo y distancia entre sí. Por el otro lado, al buscar CRISPR conocidos en un genoma, se utilizan DR ya documentados como guías de búsqueda. Detectar nuevos CRISPR en metagenomas es de utilidad, pues la información contenida en los *spacers* permite estudiar la interacción entre virus y microbios en determinados entornos.

El procesamiento computacional de genomas requiere un gran uso de recursos debido al tamaño de los archivos genómicos. No solo es necesario contar con suficiente memoria para abrir los archivos, también debe ser posible realizar consultas complejas sobre estos. Este requerimiento de memoria por sí solo no sería tan problemático si no implicara acceder a disco para resolver consultas interesantes. Los accesos a disco ralentizan los procesos computacionales, por lo que es favorable evitarlos y operar lo más posible en RAM. Una forma de mejorar el rendimiento de procesos es utilizar estructuras de datos compactas cuando es posible, lo que reduce significativamente la memoria utilizada y permite trabajar en RAM.

Se han secuenciado microbios del Desierto de Atacama con el propósito de estudiar su genética, incluyendo posibles CRISPR y *spacers*. Debido a que este metagenoma no ha sido estudiado aún, es necesario realizar una extracción *de novo* de CRISPR cuya efectividad sea alta.

El Desierto de Atacama es un caso de estudio interesante para investigadores de organismos extremófilos, i.e. organismos que habitan en condiciones fundamentalmente distintas a la mayoría de los organismos de la Tierra. Las altas temperaturas, completa aridez y elevada radiación del desierto llevaron a creer que este lugar no era posible que existieran seres vivientes. Recientemente se han descubierto microbios viviendo en este ecosistema, y su estudio es de particular interés porque permitiría identificar estrategias para mitigar el estrés hídrico y la radiación ultravioleta. En cuanto a CRISPR, el estudio de las separaciones entre repeticiones otorgaría una visión sobre las características que podrían tener virus si es que estos fueran encontrados en el planeta Marte, dado a la similitud entre este planeta y el desierto.

1.3. Objetivos

Objetivo General

Este trabajo tiene como objetivo determinar si el rendimiento de algoritmos detectores de CRISPR *de novo* en genomas puede mejorar con el uso de estructuras de datos compactas. Además, se desea generar anotaciones de CRISPR *de novo* en metagenomas de bacterias no documentadas del Desierto de Atacama. Se diseñará, implementará y evaluará una nueva estrategia para encontrar secuencias tipo CRISPR en metagenomas. Esta herramienta se comparará con el software existente descrito en la Sección 2.3 en cuanto a la cantidad y calidad de las secuencias encontradas y su rendimiento.

Objetivos Específicos

1. Desarrollar un nuevo algoritmo de detección de CRISPR *de novo* que utilice árboles de sufijos compactos y árboles wavelet.

2. Evaluar calidad de solución midiendo cuántos CRISPR conocidos reconoce en genomas ya documentados.

3. Medir rendimiento de la herramienta en cuanto a tiempo de ejecución y uso de memoria. Luego comparar el rendimiento con el de las herramientas existentes.

4. Aplicar los programas sobre metagenomas de bacterias de Desierto de Atacama y comparar los CRISPR reportados.

1.4. Metodología

La metodología del trabajo se dividió en cinco fases, las cuales se describen en grandes rasgos a continuación.

- Investigación: Esta parte del desarrollo del trabajo se dedicó al estudio de estructuras de datos compactas, sus alcances y limitaciones. También se estudió sobre posibles librerías que faciliten el trabajo a desarrollar, sobre todo en lo que respecta a wavelet trees. En esta etapa del trabajo también se estudiaron soluciones existentes al problema de detección de CRISPR a modo de definir el estado del arte en esta área. Además se analizaron problemas de stringología análogos al abordado en este trabajo, junto a sus posibles soluciones. Asimismo se investigó acerca de material genético y CRISPR, consultando con expertas y expertos en el área de genética de bacterias extremófilas para así obtener un mejor entendimiento del problema y los datos con los que se trabajaría.

- Análisis y Diseño: En esta etapa se decidió el esquema general de la solución, en base a lo aprendido en la fase anterior. También se tomaron decisiones en cuanto a cómo se

validaría y compararía el algoritmo una vez finalizado.

- Desarrollo: Se implementó la solución ideada usando el lenguaje de programación C++ y una librería dedicada al trabajo con estructuras de datos compactas.

- Validación: Para esta fase del trabajo fue crucial obtener datos de prueba, en el caso de este trabajo estos corresponden a secuencias de genomas de bacterias y microorganismos cuyos CRISPR ya han sido estudiados. Se probó el funcionamiento del algoritmo usando datos *mock*, y una vez finalizada la implementación se corrieron experimentos con distintos genomas para medir la calidad de la solución en cuanto a resultados obtenidos y uso de recursos. Estos experimentos se replicaron en una selección de software de detección de CRISPR con el objetivo de comparar rendimientos.

- Aplicación: Se utilizó la herramienta implementada para detectar CRISPR en genomas de microbios del Desierto de Atacama, ensamblados a partir de la secuenciación de un metagenoma obtenido del desierto.

Capítulo 2

Marco Teórico

En este capítulo se exponen los elementos relevantes para la completa comprensión de este trabajo. Primero se definen las estructuras de datos compactas utilizadas, luego se comparten nociones básicas sobre CRISPR y bioinformática y finalmente se habla sobre herramientas de detección de CRISPR existentes.

2.1. Estructuras de Datos Compactas

Las estructuras de datos compactas permiten representar objetos en espacio significativamente reducido con respecto a las estructuras tradicionales, manteniendo sus funcionalidades. Esto otorga la posibilidad de operar en memoria principal sobre estructuras que tradicionalmente requerirían uso de memoria secundaria y costosos accesos a disco.

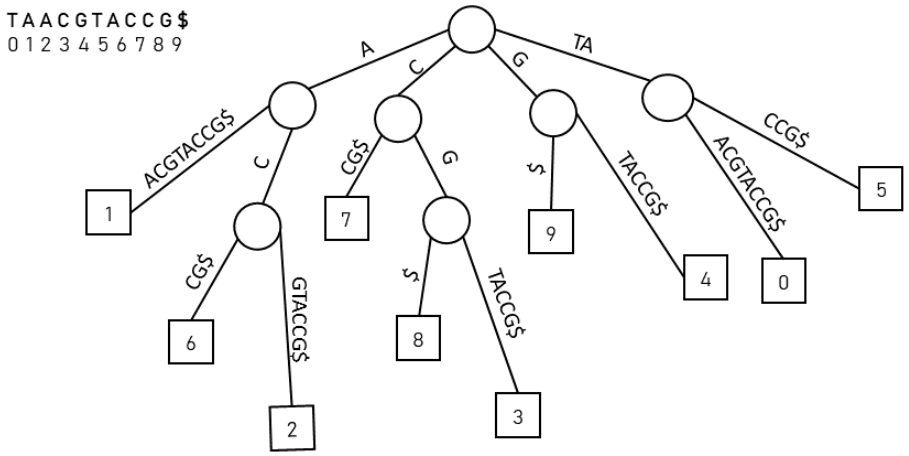
2.1.1. Árboles de Sufijos Compactos

Un árbol de sufijos es una estructura de datos creada a partir de un texto que almacena todos los sufijos del texto y su ubicación en este. Los sufijos de un texto son el conjunto de *substrings* que comienzan en cada posición del texto y se extienden hasta su final, por lo que un texto de largo n tiene $n+1$ sufijos si contamos el *string* vacío. Cada hoja del árbol de sufijos de un texto T almacena la ubicación en el texto original de un sufijo; las aristas están etiquetadas con una subcadena de T, dos aristas salientes del mismo nodo no pueden compartir un prefijo. De este modo, para cada hoja, la concatenación de etiquetas del camino trazado desde la raíz hasta la hoja produce el sufijo del texto ubicado en la posición almacenada en la hoja. El largo del texto producido al concatenar el camino hasta un nodo corresponde a la profundidad de texto de dicho nodo.

Otra representación de esta información, que ocupa aun menos espacio, es un arreglo de sufijos. Este es un arreglo A cuyo elemento A[i] apunta al i-ésimo sufijo en orden lexicográfico. Este ahorro de espacio viene con un aumento en el costo de algunas operaciones, por lo que

dependiendo del caso será más provechoso usar una u la otra estructura.

En la Figura 2.1 se muestra el árbol y correspondiente arreglo de sufijos generados a partir de la palabra “TAACGTACCG”. En el arreglo de sufijos se guardan los índices correspondientes a los sufijos ordenados lexicográficamente.



i	0	1	2	3	4	5	6	7	8	9
SA	**1**	**6**	**2**	**7**	**8**	**3**	**9**	**4**	**0**	**5**

Figura 2.1: Árbol y arreglo de sufijos de la palabra "TAACGTACCG", el símbolo ’$’ indica el fin de la palabra.

Las implementaciones clásicas de árboles de sufijos utilizan $O(n \log n)$ bits, con n siendo el largo del texto. Sin embargo, un árbol de sufijos comprimido utiliza $O(n \log \sigma)$ bits, donde σ representa el tamaño del alfabeto [16]. Esto es particularmente útil en aplicaciones a problemas de ADN, puesto que el alfabeto de las secuencias es de tamaño 4, considerablemente más pequeño que el largo de los genomas de bacterias, que ronda en los millones [19].

Estas estructuras tienen variados usos en algoritmos de procesamiento de textos, un uso de un árbol de sufijos es la búsqueda de patrones de texto repetidos. Para cualquier nodo interno del árbol, la cantidad de hojas a las que se puede llegar desde ese nodo equivale al número de veces que se repite la palabra obtenida al concatenar las etiquetas de la ruta desde la raíz a dicho nodo. Por ejemplo en la Figura 2.1 se puede ver que el número de hojas a las que se llega siguiendo la ruta $raiz \rightarrow C \rightarrow G$ corresponde con la cantidad de veces que la secuencia “CG” aparece en el texto original: 2.

2.1.2. Árboles Wavelet

Un árbol *wavelet* es una estructura de datos que permite almacenar y hacer consultas sobre una secuencia de caracteres de manera eficiente. Cada nodo del árbol contiene un mapa de bits representante de una subsecuencia del texto y se construye recursivamente,

particionando el alfabeto de la secuencia del nodo en dos en cada paso; si L es el elemento mínimo del alfabeto y R el máximo, entonces una partición contendrá los símbolos entre L y la mediana del alfabeto, mientras que la otra corresponde a la segunda mitad. En el mapa de bits del nodo se indica a qué partición del alfabeto pertenece cada carácter, colocando un 0 si es que corresponde a la primera mitad del alfabeto y un 1 de lo contrario. Cada símbolo de la secuencia se inserta en el hijo izquierdo o derecho del nodo, dependiendo de a qué partición corresponda. Esto se repite hasta obtener un set de hojas que contienen un símbolo individual del alfabeto.

La Figura 2.2 ilustra el árbol *wavelet* de la secuencia "TAACGTACCG". La estructura solo almacena los mapas de bit de cada nodo y la relación entre los nodos del árbol, pero en la imagen se muestran los textos representados para una mejor comprensión.

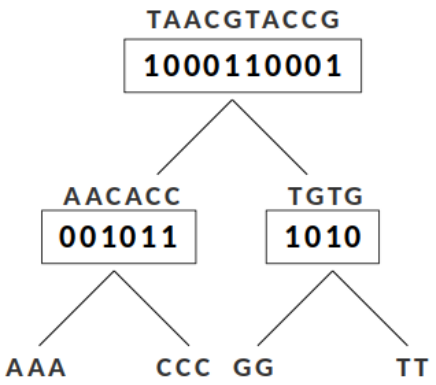


Figura 2.2: Árbol wavelet de la palabra "TAACGTACCG".

Dos operaciones básicas de un bitmap son $rank_0(B, \mathrm{i})$ y $select_0(B, \mathrm{i})$. La primera retorna la cantidad de 0 en el bitmap B hasta la poisición i, mientras que la segunda encuentra el i-ésimo 0 en el bitmap. Los árboles wavelet permiten rastrear símbolos, es decir obtener el i-ésimo elemento de la secuencia original, usando la operación *rank(B, i)* recursivamente hasta llegar al bitmap de mayor profundidad, del cual se deduce la etiqueta de la hoja y por consiguiente el elemento consultado.

Por ejemplo, si se desea conocer el cuarto carácter de la secuencia "TAACGTACCG", cuyo árbol wavelet se muestra en la Figura 2.2, primero se obtiene el cuarto elemento del mapa de bits de la raíz para saber a qué partición y subárbol corresponde el elemento. En este caso es 0, lo que significa que el elemento se encuentra en la partición $[A, C]$ del alfabeto. Se realiza $rank_0(root, 4)$ para obtener la posición del carácter buscado en el subárbol izquierdo. Como el carácter corresponde al tercer 0 del mapa de la raíz, entonces se accede al tercer elemento del mapa de bits correspondiente. Este número corresponde a un 1, lo que indica que el elemento se encuentra en la segunda mitad de la partición del alfabeto de ese nodo. Esta partición solo contiene un carácter, de lo que se obtiene que el elemento buscando corresponde a "C".

Los árboles wavelet pueden ser usados para representar grillas de puntos [12] construyendo el árbol a partir de la secuencia obtenida al ordenar las coordenadas del eje Y según el eje

X. De este modo la posición en el arreglo indica la coordenada X del punto representado, mientras que el valor guardado es la coordenada Y. Las grillas representadas típicamente guardan n puntos en un espacio de $n \times n$, de manera tal que solo haya un punto por fila y por columna. Esta forma de representar grillas permite contar y reportar la cantidad de puntos que residen dentro de un rectángulo de la grilla en órdenes logarítmicos.

Dado un valor y un rango de valores, es posible obtener el siguiente elemento de la secuencia guardada en un árbol wavelet que sea mayor que el valor entregado y se encuentre dentro del rango pedido, a través de la consulta *range_ next_ value(S, i, j, x)* presentada por Gagie, Navarro y Puglisi [4]. Esta operación retorna el menor elemento $S[r] > x$ del árbol tal que $i < r < j$.

Esta consulta funciona recorriendo el árbol wavelet recursivamente, partiendo desde la raíz. Si es que x corresponde a una hoja que se encuentra a la derecha de la raíz, se continúa recursivamente por el hijo derecho, ajustando el rango en el cual se puede encontrar el valor en el bitmap del hijo. Esto se hace actualizando los valores de i y j como la cantidad de "1" que se encuentran en el bitmap de la raíz hasta las posiciones i y j, respectivamente usando la operación $rank_1$. En el caso que x sea representado por una hoja a la izquierda de la raíz, se desciende al hijo izquierdo, nuevamente actualizando los valores del rango pero esta vez utilizando $rank_0$ para contabilizar la cantidad de "0" en el bitmap de la raíz desde el inicio hasta i y j. Si se llega a una hoja, se retorna $j - 1$, valor correspondiente a la posición del elemento buscado. Si el intervalo $[i, j]$ se vuelve vacío, se retorna nulo. Lo mismo ocurre cuando la recursión proveniente de un hijo derecho retorna nulo, pues significa que no hay ningún valor mayor a x en el rango indicado. En cambio, si es que un hijo izquierdo retorna nulo, todavía existe la posibilidad de que exista un valor en el hijo derecho que sea mayor a x y se encuentre en el rango de posiciones indicado. Por esto se desciende por el hijo derecho, actualizando los valores de i y j. Si es que el rango no resulta vacío, el valor que se busca corresponde a la hoja más a la izquierda del nodo.

Una grilla conectando las posiciones lexicográficas con las posiciones en el texto de todos los sufijos permitirá encontrar todos los candidatos a secuencias CRISPR con suficientes repeticiones usando el árbol de sufijos y luego verificar que satisfagan los requisitos de cercanía usando el árbol wavelet.

2.2. CRISPR y bioinformática

Como se enunció en la Introducción de este trabajo, un *locus* de CRISPR consiste en una serie de secuencias repetidas periódicamente denominadas *direct repeat* o DR, separadas por cadenas de nucleótidos llamadas *spacers*. El largo de un DR debe estar entre los 24 y 48 pares de bases nitrogenadas (unidad deominada bp) [10], i.e. caracteres. Asimismo, los *spacers* en un arreglo CRISPR tienen un largo de entre 28 y 37 bp. Generalmente los *spacers* de un mismo CRISPR tienen un largo similar entre sí.

En cuanto a las repeticiones, no hay un mínimo ni un máximo de ocurrencias que se deben presentar para que una secuencia de ADN sea considerada CRISPR. Si es que dos

ocurrencias cumplen con los requisitos de largo y distancia entre sí, ese fragmento es un CRISPR. Tampoco hay un patrón característico de CRISPR, el contenido de un DR puede ser cualquiera mientras cumpla con las condiciones expuestas. Mientras que los DR de un CRISPR pueden ser iguales a los de otro, los *spacers* deben ser secuencias distintas entre sí hasta dentro de un mismo CRISPR.

El largo de un genoma bacteriano se encuentra en el rango de 130.000bp y 10.000.000bp, con un promedio de 3.87 millones de pares de bases.[2]

Las secuencias de ADN suelen almacenarse en formato `FASTA`; este guarda un número variable de registros, y por cada registro de secuencias guarda la secuencia en sí y un identificador. El registro comienza con una línea de encabezado cuyo primer carácter es `">"`seguido del identificador de la secuencia. Las líneas siguientes contienen la secuencia.

>Mus_musculus_tRNA-Ala-AGC-1-1 (chr13.trna34-AlaAGC)
GGGGGTGTAGCTCAGTGGTAGAGCGCGTGCTTAGCATGCACGAGGCCCTGGGTT
CGATCCCCAGCACCTCCA

Extracto de secuencia en formato FASTA

2.3. Trabajo Relacionado

Existen variadas herramientas capaces de detectar CRISPR en genoma de bacterias que usan distintos enfoques para lograrlo. Tres de las más prominentes se describen a continuación. CRISPR Recognition Tool (CRT) [1] tiene un enfoque de búsqueda secuencial, en la cual se encuentran candidatos a CRISPR que luego son verificados. Otra herramienta es pilercr [3], que utiliza matrices de autosimilitud para encontrar repeticiones locales, es decir, dos repeticiones contiguas. Con esta información se construye un grafo en el cual cada componente conexa representa un candidato a CRISPR que posteriormente es comprobado. Finalmente, CRISPRFinder [13] es una herramienta web que utiliza un programa basado en arreglos de sufijos para encontrar candidatos a CRISPR. Estos candidatos luego son filtrados en base a distintos criterios, descartando los que no cumplen con todas las características que posee un CRISPR.

Si bien las herramientas anteriores son buenas por sí solas para encontrar secuencias CRISPR, es recomendable utilizar más de una a la vez para complementar los resultados puesto que las herramientas por separado no logran reportar todas las secuencias [15].

Como se mencionó en la Introducción de este documento, el metagenoma es el material genético de muestras obtenidas del ambiente. La naturaleza de los métodos de secuenciación y ensamblaje de ADN dificultan el proceso de detección de CRISPR en metagenomas con los métodos presentados [14]. Se han desarrollado algunas herramientas enfocadas específicamente en detectar CRISPR en secuenciaciones metagenómicas.

CRASS es una herramienta diseñada para enfrentar este problema [17]. Funciona identificando secuencias CRISPR aprovechando la estructura repetitiva de estas, adaptando el

método propuesto en CRT. Luego se generan cúmulos de DR y se utiliza un algoritmo basado en grafos para encontrar el arreglo de *spacers* de cada CRISPR. Estos arreglos deben pasarse manualmente por un programa externo para obtener la totalidad de las secuencias CRISPR.

Otra propuesta es metaCRISPR [11], que plantea un método completamente automático de identificación de CRISPR en metagenomas. Al igual que CRASS, filtra las secuencias que pueden contener CRISPR basándose en la identificación de repeticiones. Luego forma cúmulos de secuencias guiándose por los traslapes de estas. Cada cúmulo es ensamblado utilizando nuevamente un algoritmo basado en grafos, generando candidatos a CRISPR. Finalmente se validan los candidatos, descartando aquellos que no satisfacen los criterios de CRISPR.

2.3.1. Exploración de Software

Se enfocó el estudio más profundo de herramientas existentes en tres de las mencionadas anteriormente: CRT, pilercr y metaCRISPR. Con el fin de familiarizarse con estos instrumentos, comprender su funcionamiento y entrega de resultados, se corrieron con genomas ya estudiados obtenidos de la base de datos CRISPRdb, que contiene información detallada sobre CRISPR conocidos en genomas. La exploración de cada software consistió en intentar correrlo con los genomas escogidos, ver el formato del output entregado y comparar los resultados entre sí y con la base de datos de manera informal. En la Tabla 2.1 se muestra la información de los CRISPR identificados y verificados en la bacteria *Acidovorax avenae subsp. avenae ATCC 19860*.

CRISPR_id	Start Position	End Position	N° of spacers	DR consensus
NC_015138_1	300343	303481	47	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
NC_015138_2	304387	304489	1	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
NC_015138_3	305395	306222	12	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
NC_015138_5	1785060	1785158	1	GTTTCAACCCACGCGCCCGCACAGGGCGCGAC
NC_015138_6	1785247	1786614	20	GTTTCAACCCACGCGCCCGCACAGGGCGCGAC
NC_015138_7	4426774	4430214	51	GTCGCGCCCCGCGTGGGCGCGTGGGTTGAAAC

Tabla 2.1: Anotación de CRISPR en *Acidovorax avenae* obtenida de CRISPRdb.

CRT

Para ejecutar CRT basta con descargar el archivo ejecutable[1] y correrlo desde la consola de comandos con el archivo de secuencias como parámetro, como se muestra a continuación.

```
java -cp CRT1.2-CLI.jar crt <archivo de input><archivo de output>
```

El archivo resultante indica el organismo analizado, la cantidad de bases nitrogenadas de la secuencia, un listado de cada CRISPR identificado incluyendo su DR, los *spacers* y las posiciones de estos dentro del genoma, y el tiempo de ejecución. En la Figura 2.3 se muestra un fragmento del archivo resultante de ejecutar CRT con el genoma de uno de los organismos. Como se puede ver en la imagen, se detalla la secuencia de la repetición y de cada separador.

[1] obtenido en `http://www.room220.com/crt/`

```
ORGANISM:  NC_015138.1 Acidovorax avenae subsp. avenae ATCC 19860, complete genome
Bases: 5482170


CRISPR 1   Range: 300343 - 303480
POSITION        REPEAT                              SPACER
--------        ----------------------------------  ----------------------------
300343          AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC    TGAGGCAGCACGCGATTTCCGCCAGCAGGT  [ 36, 30 ]
300409          AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC    GCTGGCAAAGTGGACCATGCTCCAACATGT  [ 36, 30 ]
300475          AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC    CCAAGGTCACCGTGGCTGTCATGTCGCTGG  [ 36, 30 ]
300541          AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC    TGCCGGTGGTCTATCGAGCGTAACGGCGCG  [ 36, 30 ]
300607          AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC    TCGGATGCTCGCGTGCATCCATCGGCGATT  [ 36, 30 ]
```

Figura 2.3: Fragmento de archivo de salida al ejecutar CRT con el genoma de la bacteria *Avidovorax avenae*.

Al comparar los resultados entregados por CRT, detallados en la Tabla 2.2, con la información en CRISPRdb, se detectan pequeñas discrepancias. La herramienta solo detectó 4 de los 6 CRISPR identificados, fallando al detectar los CRISPR cortos compuestos por solo dos repeticiones y un separador.

CRISPR	Start Position	End Position	N° of Spacers	Direct Repeat Consensus
1	300343	303480	47	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
2	305395	306222	12	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
3	1785247	1786614	20	GTTTCTATCCACGCGCCCCGCATGGGGCGCGAC
4	4426774	4430214	51	GTCGCGCCCCGCGTGGGCGCGTGGGTTGAAAC

Tabla 2.2: Resultado de ejecutar CRT con genoma de *Avidovorax avenae*.

pilercr

La ejecución de pilercr es igual de sencilla, se bajaron los archivos necesarios[2] y se corrió el programa desde la consola de comandos:

```
./pilercr -in <archivo de input>-out <archivo de output>
```

El reporte entregado en el archivo de salida es más detallado que en el caso de CRT y contiene información sobre el uso de memoria durante el proceso. Por cada arreglo CRISPR se crea una tabla que incluye la posición de cada repetición en la secuencia, el largo del DR, el porcentaje de similitud con el DR consensuado, el largo del siguiente *spacer*, las 10 bases a la izquierda de la repetición, la secuencia del DR (indica si difiere del consenso) y la secuencia del *spacer*. Un ejemplo de esta tabla se puede ver en la Figura 2.4, notar que para la última repetición bajo el campo "Spacer"se indican las siguientes 10 bases de la secuencia de ADN.

[2]Los archivos pueden descargarse en este sitio `http://www.drive5.com/pilercr/`

```
Array 2
>NC_015138.1 Acidovorax avenae subsp. avenae ATCC 19860, complete genome

      Pos  Repeat     %id  Spacer  Left flank    Repeat                                Spacer
==========  ======  ======  ======  ==========    ====================================  ======
   305395      36   100.0      31  ATGACAGCGC    ....................................  GACGGCGAGTATCTAGGCAAAGTGGTACGTT
   305462      36   100.0      29  GTGGTACGTT    ....................................  TCATACAAAAAGTTTTTCTGTCATACAAA
   305527      36   100.0      30  GTCATACAAA    ....................................  CGGTTTCAATCGCATCGTTTTCGTAGCCAG
   305593      36   100.0      30  TCGTAGCCAG    ....................................  CGAAGCTCGCGAAGGGGCAGCGCTGGGGTC
   305659      36   100.0      30  CGCTGGGGTC    ....................................  TTATTTCCTCAGTGCAGGCGGGCGAATGGT
   305725      36   100.0      30  GGCGAATGGT    ....................................  AGCGGTTTTACAGCCCAGAGGAGCAGCGAG
   305791      36   100.0      30  GAGCAGCGAG    ....................................  TTCGCGTGCGCGGGGACTCAATGACAGCGC
   305857      36   100.0      30  ATGACAGCGC    ....................................  GCGCCGCTGCCGCCATTGCCCACGCCATCG
   305923      36   100.0      30  CACGCCATCG    ....................................  AGCGGTTTTACAGCCCAGAGGAGCAGCGAG
   305989      36   100.0      30  GAGCAGCGAG    ....................................  GAGGTTACTGGCAAGTGCTCGGAGGGGTGA
   306055      36    97.2      30  GGAGGGGTGA    .....................A..............  TCTATGGTTTCTCAGGCAGTAAAGAAGTCG
   306121      36   100.0          AAAGAAGTCG    ....................................  TGTACCAGCG
==========  ======  ======  ======  ==========    ====================================
       12      36              30                AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
```

Figura 2.4: Fragmento de archivo de salida al ejecutar pilercr con el genoma de la bacteria *Avidovorax avenae* correspondiente a un arreglo CRISPR.

El archivo de salida creado por pilercr también incluye tablas de resumen que indican las posiciones de cada arreglo CRISPR, el largo, la cantidad de repeticiones que contiene, el largo promedio de la secuencia repetida y espaciadora y el DR consensuado entre cada repetición. En la Tabla 2.3 se resumen los hallazgos de pilercr al ejecutarse sobre el genoma de *Avidovorax avenae*.

CRISPR_id	Start Position	End Position	Number of spacers	DR consensus
1	302259	303481	19	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
2	305395	306157	12	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
3	1785448	1786615	18	GTTTCAACCCACGCGCCCGCACAGGGCGCGAC
4	4426774	4429004	34	GTCGCGCCCCGCGTGGGCGCGTGGGTTGAAAC
5	4429181	4430215	16	GTCGCGCCCCGCGTGGGCGCGTGGGTTGAAAC

Tabla 2.3: Resultado de ejecutar pilercr con genoma de *Avidovorax avenae*.

Si bien pilercr detecta 5 arreglos de CRISPR, al ver los resultados con atención se puede notar que difieren de la información de la base de datos. Por ejemplo el último arreglo anotado en la Tabla 2.1 aparece como dos arreglos distintos en los resultados. La herramienta fue capaz de detectar correctamente los DR de cada arreglo de CRISPR, pero no su inicio y fin.

metaCRISPR

El proceso de ejecución de metaCRISPR [3] se divide en tres: la identificación de los fragmentos que pueden contener CRISPR, el clustering de los *reads* filtrados, y finalmente el ensamblaje de cada arreglo. En la primera etapa se pueden agregar opciones de ejecución como el largo mínimo y/o máximo de una repetición o espaciador, la cantidad de hilos de ejecución a usar y el máximo error que puede haber entre repeticiones.

Al indagar más en el funcionamiento de metraCRISPR se descubrió que la herramienta trabaja a partir del genoma no ensamblado, es decir su input son *reads*[4] de ADN en modo

[3]El código fuente de la herramienta se encuentra en `https://github.com/hangelwen/metaCRISPR`

[4]Fragmentos cortos de ADN que luego son montados para generar la secuencia entera.

paired-end. La secuenciación *paired-end* es un método en el cual se secuencian ambos extremos de un fragmento, que resulta en una mejor alineación de las lecturas y por ende un ensamblaje más preciso. Esta técnica produce dos archivos de lecturas, una por cada extremo. Esto implica que no será posible probar el funcionamiento de esta herramienta con el algoritmo propuesto en este trabajo, puesto que este último trabaja con los genomas ya ensamblados.

Capítulo 3

Diseño de la Solución

Durante esta etapa del trabajo se definió bien el problema y se ponderaron distintas propuestas de solución, en base al estudio de problemas análogos en stringología y las operaciones posibles con estructuras de datos compactas. A continuación se dará una visión general del punto de partida del proceso de diseño, se describirán brevemente distintas alternativas consideradas y finalmente se entrará en mayor detalle sobre la ideación de cada parte del algoritmo.

3.1. Planteamiento del Problema

La detección de un *locus* de CRISPR en una secuencia de ADN equivale a encontrar todos los substrings del texto que consistan en un patrón repetitivo (correspondiente al DR) de cierto largo tal que cada ocurrencia esté separada por una secuencia de otro tamaño determinado (los *spacers*). Estos fragmentos de texto no deben solaparse entre sí y sus repeticiones deben ser maximales dentro del rango de largo requerido, es decir, incluir la mayor cantidad de caracteres posibles manteniendo la condición de largo. Luego el problema a resolver en este trabajo es el siguiente: dado un rango de largo $[r_1, r_2]$,un rango $[s_1, s_2]$ y un texto, encontrar todos los substrings del texto tal que un fragmento de texto de largo $r \in [r_1, r_2]$ se repita de manera tal que la distancia entre el fin de una ocurrencia del patrón y el comienzo de la siguiente se encuentre en el rango $[s_1, s_2]$. Por cada conjunto de ocurrencias y sus separaciones se deberá reportar la posición en el texto de la primera ocurrencia, la cantidad de ocurrencias contenidas en el substring, y el largo del patrón repetido.

Se decidió abordar el problema con la idea general de utilizar un árbol de sufijos para detectar patrones de texto repetidos en la secuencia de ADN, junto a alguna heurística de selección de candidatos de CRISPR. Esto puesto que la topología de los árboles de sufijos permiten detectar repeticiones de patrones de manera eficiente y sencilla, como fue expuesto en la sección 2.1.1. Un árbol de sufijos permite encontrar todos los fragmentos que se repiten en un texto, pero la mera característica repetitiva de un patrón no garantiza que sea parte de un CRISPR, por lo que se requiere de una metodología para filtrar estas repeticiones y así elegir aquellas que cumplan con los requisitos de un CRISPR.

3.2. Alternativas Consideradas

El problema de encontrar CRISPR en un genoma puede verse desde un punto de vista de stringología. Un símil a este problema es el de encontrar *gapped repeats* en texto. Esto es, dado un texto, encontrar patrones dentro de este que aparezcan una cantidad mínima de veces, separados por una cierta distancia [5].

Pawel et al. [5] exponen un método para encontrar todos los *gapped repeats* maximales en un texto, correspondientes a aquellas palabras $w = uvu$ tal que ambos segmentos u no puede ser extendidos simultáneamente por un carácter por su lado izquierdo o derecho. Inicialmente se consideró utilizar el algoritmo propuesto como filtro para obtener secuencias candidatas a CRISPR, pero al indagar más en este se descubrió que su método no permite imponer condiciones al largo de los fragmentos repetidos ni sus separadores (*gaps*). Como se expone en la sección 2.2, una característica fundamental de CRISPRs es el largo de tanto sus DR como de sus *spacers*, por lo que se determinó que no era posible utilizar este algoritmo y se prefirió explorar otros acercamientos por sobre adaptar esta alternativa al problema abordado.

Al comienzo de este trabajo se creía erróneamente que una condición para que una secuencia fuera CRISPR era tener cierto número mínimo de ocurrencias de un DR, por lo que se indagó en ideas de filtro que tomaran esto en consideración. Una idea explorada fue la de por cada patrón repetido, primero contar si cumplía con tener más de k ocurrencias, siendo k el requisito de CRISPR. Esto se haría con una "ventana"de lectura de largo $k \times (r + s)$, con r siendo el rango de largo de un DR y s el correspondiente a *spacers*, e intentar leer las primeras k repeticiones con la ventana. Si es que no hay suficientes DR en la ventana, entonces se saltaría hasta la siguiente ocurrencia y se intentaría nuevamente desde ahí. En el caso de sí haber más de k repeticiones en el rango, se procedería al siguiente paso de la verificación de CRISPR. Si bien este filtro permitiría reducir bastante la cantidad de candidatos a CRISPR antes del proceso más costoso de verificar distancias entre puntos, al no ser requisito para ser considerado como CRISPR utilizar este filtro ignoraría CRISPRs erróneamente.

Se optó en vez por explorar un enfoque geométrico al problema de filtrar candidatos y verificar si es que cumplen las condiciones requeridas para ser considerados como CRISPR. Esto puesto a que estas condiciones están relacionadas al posicionamiento, largo y espaciado de los DR y sus *spacers*, por lo que es posible proyectar este problema a un espacio geométrico y utilizar soluciones existentes y eficientes de esta área para resolverlo.

3.3. Algoritmo Propuesto

El algoritmo propuesto se puede dividir en tres procesos: la selección de candidatos a CRISPR a partir del genoma, la verificación de aquellos escogidos usando las características de un CRISPR, y un filtro final de candidatos. En las secciones siguientes se describe el diseño de cada etapa del algoritmo.

3.3.1. Selección de Candidatos

Como se enunció anteriormente, la selección inicial de candidatos se hace utilizando un árbol de sufijos compacto. Luego de crear el árbol de sufijos correspondiente a la secuencia de ADN, se extraen todos los DR posibles de él. Esto se hace seleccionando la totalidad de los patrones del texto que se repitan al menos dos veces y cuya profundidad de texto se encuentre entre el rango requerido. Este paso es posible ya que todo nodo interno del árbol de sufijos representa un string que se repite al menos una vez en el texto, como se explica en la Sección 2.1.1. La Figura 3.1 ilustra un esquema de la selección de un candidato en un árbol de sufijos. El triángulo mayor representa el árbol entero, y la línea morada bosqueja una ruta trazada en el árbol, representante de un *string* que se repite varias veces en el texto original. Las líneas punteadas indican el rango de profundidad requerido para que un patrón sea considerado como candidato a DR, con r_1 siendo la profundidad mínima y r_2 la máxima. El triángulo menor describe las ramas salientes del nodo final del patrón repetido, las hojas terminales de estas ramas contienen las posiciones de cada ocurrencia del patrón en el texto.

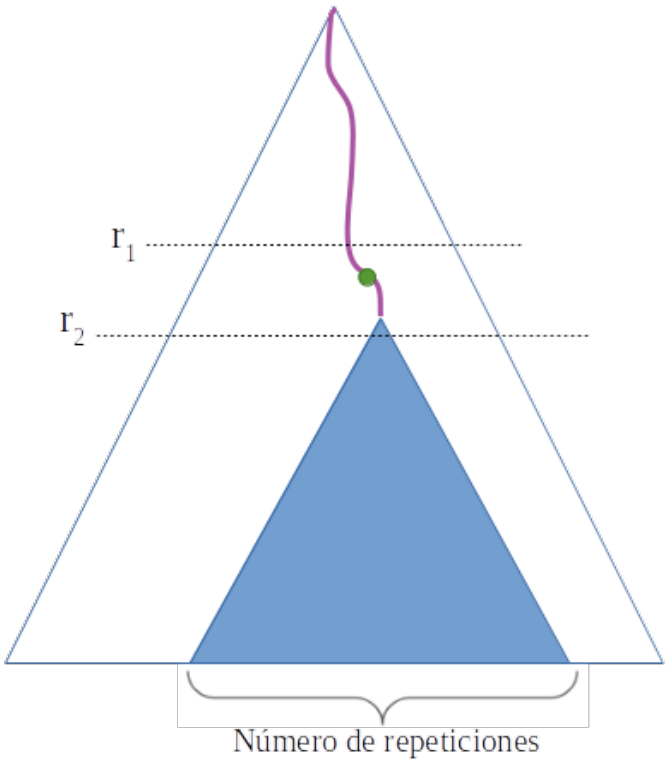


Figura 3.1: Diagrama de selección de posibles DR en un árbol de sufijos. El punto verde representa un patrón en el texto que cumple con los criterios para ser seleccionado como candidato a DR.

Por cada posible DR se pueden obtener las posiciones en el texto donde comienza cada ocurrencia al acceder a sus hojas, que guardan la ubicación en el arreglo de sufijos de cada sufijo. Es más, basta con obtener el valor de la hoja más a la izquierda del subárbol y el de la más a la derecha. Estos valores, de aquí en adelante *lb* y *rb* respectivamente, indican el

segmento del arreglo de sufijos que contiene la posición de todas las ocurrencias del texto repetido.

3.3.2. Verificación de Candidatos

Si bien los candidatos seleccionados cumplen con el largo indicado y tienen más de dos ocurrencias en el texto, estas pueden encontrarse repartidas en él y no ser parte de un CRISPR. En esta etapa del algoritmo se agrupan y seleccionan ocurrencias contiguas separadas por una distancia de entre s_1 y s_2 pares de bases nitrogenadas, formando así candidatos a CRISPR que cumplen con todos los requisitos.

Se propone formar una grilla de puntos que distribuya a los sufijos según su posición lexicográfica en un eje y su posición en el texto en el otro, como se muestra en la Figura 3.2. El resultado es una grilla que presenta un solo punto por fila y por columna, que puede usarse para comprobar la distancia de separación entre dos candidatos.

	A	T	C	G	T	A	C	G
i	0	1	2	3	4	5	6	7
SA	5	0	6	2	7	3	4	1

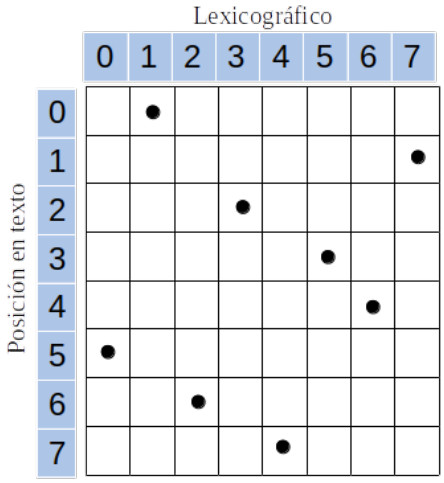


Figura 3.2: Grilla representante del arreglo de sufijos correspondiente a la palabra "ATCGTACG". Cada punto es un sufijo, su ubicación en un eje muestra su posición en el arreglo de sufijos mientras que el otro eje indica su posición en el texto original. Por ejemplo el punto $(5, 3)$ representa al sufijo "GTACG".

Esta representación permite reducir el problema de verificar el largo de los *spacers* entre candidatos al de reportar puntos en un área de una grilla. Como la siguiente ocurrencia tiene restricciones tanto en su ubicación en el arreglo de sufijos como en el texto original, es posible restringir el área de la grilla en la cual se debe encontrar este punto. Si se tiene un candidato seleccionado en el paso previo de largo l y que se encuentra en la posición i del arreglo de

sufijos SA, para comprobar si es parte o no de un CRISPR y encontrar al siguiente DR en tal caso, bastaría con verificar que en cierta área de la grilla se reporten puntos pertenecientes a los candidatos posibles. Esta área corresponde al rectángulo formado por la intersección entre los sufijos que se encuentran dentro de las posiciones $SA[i] + l + s_1$ y $SA[i] + l + s_2$[1] del texto y aquellos cuya ubicación en el arreglo de sufijos se encuentra entre el rango obtenido en el paso anterior. De este modo se verifica que la distancia entre el siguiente DR posible y el actual esté dentro del rango requerido y que ambos strings comparten como prefijo el texto repetido. En la Figura 3.3 se ilustra un ejemplo en el cual el candidato analizado se marca con un círculo rosa, el área azul marca las coordenadas que cumplen con la condición de comenzar con el mismo patrón que el candidato en cuestión, mientras que el rectángulo rosa indica aquellos sufijos que están a la distancia adecuada del candidato. La intersección de estos dos rectángulos contiene los puntos que cumplen los criterios necesarios para ser el siguiente DR en la cadena.

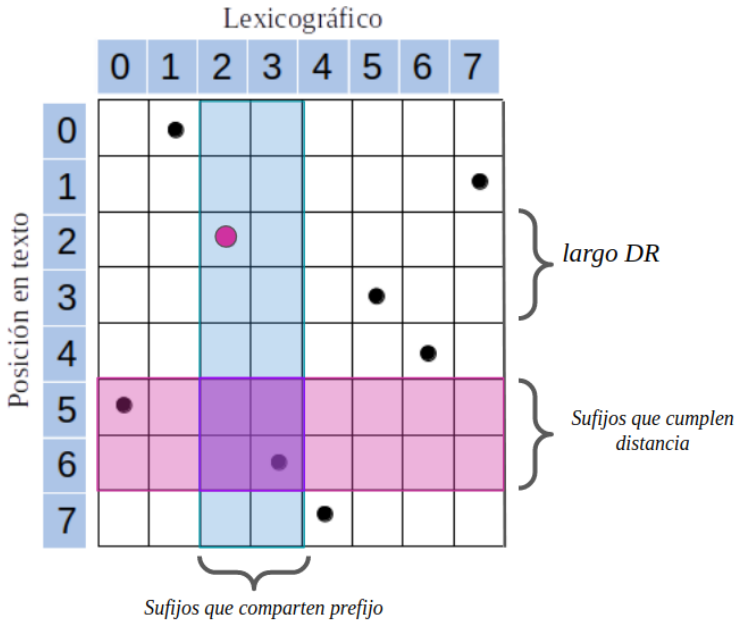


Figura 3.3: En este ejemplo el sufijo cuyo prefijo es candidato a DR se encuentra en la posición 2 del texto y en la 2 del arreglo de sufijos, el patrón repetido es de largo 2, los sufijos que comienzan con este mismo patrón se encuentran en las posiciones [2, 3] del arreglo de sufijos y el rango de largo de un *spacer* es [1, 2]. Esto restringe el área donde se debe encontrar la siguiente ocurrencia del patrón a la marcada por un rectángulo púrpura, por lo que el siguiente DR en el CRISPR se encuentra en la posición 6 del texto.

Como se vio en la Sección 2.1.2, el problema de reportar puntos en un área de una grilla puede ser resuelto usando un árbol wavelet que la represente. Dado lo expuesto, el proceso de verificación de candidatos del algoritmo consiste en la creación de un árbol wavelet a partir del arreglo de sufijos del genoma y usar la operación *range_next_value* de este árbol para comprobar la distancia entre candidatos. Por cada ocurrencia de un patrón repetido, se busca entre el grupo de ocurrencias la siguiente que esté dentro del rango adecuado. Si es que

[1]Siendo s_1 y s_2 el rango del largo que deben tener los *spacers*.

cumplen con este requisito, entonces forman parte de un CRISPR y deben ser agrupadas. De este modo se obtienen todas las cadenas que forman CRISPR en el genoma.

3.3.3. Filtro

La metodología propuesta no asegura que los patrones repetidos sean maximales, por lo que todos los substrings de un patrón de texto que cumplan con el requisito de largo y de separación son seleccionados como CRISPR. Para subsanar este problema se optó por realizar un proceso de filtrado final que remueve las cadenas que se encuentren contenidas dentro de otras, a través de la comparación de sus posiciones iniciales y finales.

Capítulo 4

Desarrollo

En este capítulo se abordará el desarrollo de la solución[1] planteada en la sección 3.3. Se expondrán más detalles sobre la implementación, aplicando el diseño propuesto y mostrando algunos casos de uso.

Primero se compartirán algunas consideraciones generales al respecto del desarrollo de la solución y luego se presentará la implementación en tres secciones, una para cada una de las etapas del algoritmo: En la sección 4.2 se hablará del proceso de selección de candidatos. Por su parte, en las secciones 4.3 y 4.4 se expondrá al respecto de la verificación de candidatos y el filtro de cadenas de CRISPR. Finalmente en la sección 4.5 se explicará el modo de uso del algoritmo.

4.1. Consideraciones Generales

Para el desarrollo de la solución se optó por utilizar el lenguaje de programación C++ debido a su eficiencia y manejo de objetos. En cuanto al manejo y creación de las estructuras, se hizo uso de una librería externa pues la implementación de estructuras de datos compactas es compleja y propensa a errores.

Se decidió utilizar la librería Succinct Data Structure Library (SDSL) [6], escrita en el lenguaje de programación C++. Esta librería provee implementaciones intuitivas de estructuras de datos compactas, incluyendo árboles wavelet y arreglos de sufijos. Además, SDSL asegura construcción eficiente de las estructuras y un uso de espacio fiel a los resultados teóricos.

[1] El código completo de este trabajo se encuentra en el repositorio Git `https://github.com/matildeRivas/crisprSDS`

4.2. Selección de Candidatos

El primer paso del algoritmo es crear el árbol de sufijos compacto a partir del genoma a analizar. Esto se hace utilizando la librería SDSL, con la cual se crea un árbol de tipo `cst_sada` a partir de un archivo de texto que contiene la secuencia. Este tipo de árbol utiliza secuencias de paréntesis balanceados para representar nodos. Se escogió esta clase de árbol por sobre otras disponibles en la librería puesto que provee operaciones navegacionales comparativamente más rápidas.

Una vez construido el árbol, se itera por todos los nodos internos de este usando el iterador de C++. Se chequea que la profundidad del string, i.e. el largo del patrón repetido, esté en el rango necesario. De no ser así, se prosigue con el siguiente nodo.

Si es que el texto sí cumple con el largo necesario, se obtiene el valor de la hoja más a la izquierda y la más a la derecha del subárbol saliente del nodo en cuestión, utilizando los métodos `lb(nodo)` y `rb(nodo)` de la librería SDSL. Estos valores corresponden a la posición en el arreglo de sufijos en la cual se encuentran los sufijos representados por esas hojas. Estos valores indican también el rango del arreglo de sufijos en los cuales se encuentran todos los sufijos cuyos prefijo es el patrón repetido, lo que permite obtener la posición de todas las ocurrencias de un patrón y la cantidad total de repeticiones. El largo del texto repetido y el rango que ocupan las ocurrencias en el arreglo de sufijos son guardados como una tupla en una lista que va almacenando los candidatos a CRISPR. Se procede de este modo hasta haber recorrido todos los nodos internos del árbol, obteniendo como resultado un listado del cual se pueden obtener la posición de cada ocurrencia de un patrón repetido que cumpla con el requisito de largo.

El Algoritmo 4.1 bosqueja este proceso mostrando el flujo de trabajo y las principales acciones de esta etapa.

Algoritmo 4.1: Selección de Candidatos

```
1  input: arbol _de_sufijos, minimo_de_repeticiones, largo_minimo, largo_maximo
2  output: list(largo de patron, posicion inicial en SA, posicion final en SA)
3  begin
4      candidatos = []
5      foreach nodo in arbol_de_sufijos:
6          if largo_minimo < nodo.profundidad_string < largo_maximo:
7              add (nodo.profundidad_string, primera ocurrencia, ultima ocurrencia)
8                  to candidatos
9      return candidatos
10 end
```

4.3. Verificación de Candidatos

Utilizando implementaciones provistas por la librería SDSL se crean el árbol wavelet y el arreglo de sufijos correspondientes al genoma a partir del archivo de entrada. El algoritmo itera por cada tupla obtenida en la etapa anterior, descartando ocurrencias que no formen parte de un CRISPR y agrupando las que sí.

Este proceso de verificación se hace pasando por todos los valores dentro del rango del arreglo de sufijos, $[lb, rb]$, que comprende las posiciones de cada ocurrencia del patrón repetido. Por cada patrón se usa la función de búsqueda por rango que provee la implementación de árboles wavelet de la librería para recorrer los puntos que se encuentran entre $[lb, rb]$ en el eje X de la grilla y $[1, n]$ por su eje Y, buscando si es que dos puntos se encuentran a distancia entre $s1$ y $s2$.

Para esto se extraen los puntos en orden del eje Y, es decir en orden de posición en el texto original. Se mantiene una cadena formada por ocurrencias continuas que cumplen con el requisito de distancia, si el punto extraído se encuentra a una distancia entre $l + s1$ y $l + s2$ del último punto almacenado en la cadena (donde l corresponde al largo de esta ocurrencia), entonces se agrega a esta. En caso contrario se almacena la cadena en un listado de CRISPR si es que es conformada por más de un punto, y se descarta si no.

Algoritmo 4.2: Verificación de Candidatos

```
input: arbol_wavelet WT, arreglo de sufijos SA, distancia_minima,
distancia_maxima, lista_candidatos
output: lista(repeticiones, largo DR, posicion inicial, posicion final)
begin
    crispr = []
    foreach grupo in lista_candidatos:
        lb = grupo.lb
        rb = gurpo.rb
        puntos = WT.range_search(lb, rb, 0, largo SA)
        cadena = puntos[1]
        for p in puntos:
            if p > cadena[ultimo] + largo + s1 and p < cadena[ultimo] + largo + s2:
                add p to cadena
            else:
                if cadena.largo > 1:
                    add cadena to crispr
                else:
                    cadena = p
        if cadena.largo > 1:
            add cadena to crispr
end
```

Una vez finalizado este recorrido se cuenta con una lista de cadenas de CRISPR. Como más de un CRISPR en un genoma pueden tener el mismo DR, en este proceso se puede formar más de una cadena de CRISPR.

Cada CRISPR es guardado en una lista en modo de una tupla conteniendo el número de repeticiones de su DR, el largo de este texto repetido, y las posiciones de su primera y última ocurrencia. Una vez finalizado el proceso de verificación esta lista contiene todos los fragmentos del genoma que cumplen con los requisitos para ser CRISPR.

El Algoritmo 4.2 presenta en forma de pseudocódigo la implementación descrita anteriormente. Notar que se utiliza el árbol wavelet para realizar la búsqueda por rango, pero por fines explicativos se optó por no incluir esto de manera explícita en el pseudocódigo.

4.4. Filtro

Como se mencionó en la Sección 3.3, el algoritmo hasta este punto incluye dentro del listado de CRISPR cadenas que están comprendidas dentro de otras. Para filtrar estas subcadenas se ordena la lista obtenida en el paso anterior por la posición de la primera ocurrencia del DR de cada CRISPR. Luego se realiza una pasada por la lista, manteniendo un candidato como "verificador". Si la posición inicial de un candidato es menor que la última posición del verificador, significa que la cadena actual y el verificador se solapan. En esta situación se descarta el candidato a menos que tenga más repeticiones de DR que el verificador. En ese caso el candidato actual pasa a ser el nuevo verificador y se prosigue con el siguiente elemento de la lista. En caso de que el candidato comience después de que el verificador termine, significa que el verificador es maximal y se mete a la lista de CRISPRs que han pasado filtro. Nuevamente el candidato se convierte en el verificador y se prosigue.

El listado final corresponde a los CRISPR encontrados en el genoma, ya que cumplen con los requisitos de largo y separación de DR y son maximales tanto en número de repeticiones como largo de DR.

4.5. Ejecución

La ejecución del algoritmo se hace a través de la línea de comandos. Primero se debe compilar el código, utilizando los comandos `cmake` y luego `make` en el directorio raiz del código. Una vez que termina la compilación, se puede correr el algoritmo como se muestra a continuación.

```
./crisprSDS <archivo de input><archivo de output>[minimo de repeticiones]
```

Al ejecutar el código se debe indicar la ruta al archivo que contiene el genoma y la ruta al archivo en el cual se desea escribir los resultados. Opcionalmente se puede definir el número mínimo de DR que debe tener un CRISPR para ser considerado como tal.

El archivo de entrada debe contener únicamente la secuencia de ADN del organismo, sin anotaciones. Tampoco deben haber saltos de línea dentro del archivo, toda la secuencia debe presentarse de forma continua. De otro modo el algoritmo no podrá reconocer las repeticiones de manera adecuada.

Una vez finalizado el algoritmo, los resultados se pueden encontrar en el archivo de salida indicado al momento de ejecutarlo. Este consiste en un archivo de texto que lista los CRISPR encontrados en el genoma incluyendo en cada línea el número de repeticiones que lo componen, el largo de su DR, la posición del primer DR y la del último DR. Estos valores se encuentran separados por tabulaciones para facilitar el análisis y lectura automatizada de los resultados.

Capítulo 5

Validación y Resultados

Este capítulo se enfoca en el proceso de validación del trabajo realizado. Acorde a los objetivos planteados para este trabajo, se evaluó el rendimiento de la solución en cuanto a la calidad de los resultados encontrados y el uso de recursos, comparándolo con el de herramientas ya existentes.

Primero se detallan los datos utilizados para realizar las pruebas, luego se comparte el diseño de los experimentos que se llevaron a cabo. Finalmente se comparten los resultados de las pruebas de calidad y de consumo de recursos.

5.1. Datos de Prueba

Para probar los instrumentos se buscaron secuencias de ADN estudiadas en CRISPRdb[1] [7], una base de datos que almacena genomas secuenciados y las secuencias CRISPR que se han encontrado y confirmado en ellos. Contiene información como la secuencia de los DR y los *spacers* de cada registro de CRISPR así como sus posiciones en el genoma y la cantidad de repeticiones. Se eligieron secuencias de seis organismos, que se descargaron en el formato necesario del sitio del Centro Nacional de Información Biotecnológica[2] de Estados Unidos (NCBI).

Los organismos fueron escogidos aleatoriamente, procurando que haya una diversidad en cuanto a cantidad de CRISPR y tamaño del archivo que contiene su secuenciación. En la Tabla 5.1 se listan los genomas utilizados, junto al tamaño del archivo y a la cantidad de CRISPR que posee el organismo. Más información respecto a los CRISPR pertenecientes a cada genoma se puede encontrar en el Apéndice A.

[1] Encontrada en https://crispr.i2bc.paris-saclay.fr/
[2] https://www.ncbi.nlm.nih.gov/

id NCBI	Nombre	Tamaño de archivo	n de CRISPR
NC_013210	Acetobacter pasteurianus	191,8 kB	1
NC_016148	Thermovirga lienii	2 mB	2
NC_013124	Acidimicrobium ferrooxidans	2,2 mB	2
NC_014392	Caldicellulosiruptor obsidiansis	2,5 mB	10
NC_006513	Aromatoleum aromaticum	4,3 mB	3
NC_015138	Acidovorax avenae subsp. avenae	5,5 mB	6

Tabla 5.1: Genomas escogidos y sus características

5.2. Diseño de Experimentos

Para evaluar la calidad de los resultados entregados por el algoritmo se ejecutó la solución con los genomas de CRISPR conocidos enunciados en la sección anterior y se compararon los resultados obtenidos con la información expuesta en CRISPRdb. Cada cadena de CRISPR resultante de la ejecución fue comparada con las secuencias CRISPR reales; si es que ambas calzan perfectamente, se considera como una detección total. En el caso de que no se trate de calces completos, ya sea porque la cadena solo comprende un fragmento del CRISPR conocido o porque su DR difiere levemente del establecido se consideró que la cadena es una detección parcial del CRISPR.

Con esto en mente se definieron las siguientes métricas de evaluación:

- Precisión: el porcentaje de resultados que corresponde a un CRISPR completo.
- Precisión Parcial: el porcentaje de resultados que corresponde a una sub-cadena de un CRISPR real.
- Recall: el porcentaje de CRISPR detectados.
- Recall Parcial: el porcentaje de CRISPR que fueron detectados parcialmente o de manera fragmentada, i.e. varias cadenas encontradas forman un CRISPR.

Los experimentos de rendimiento se realizaron ejecutando las tres soluciones y midiendo el tiempo de ejecución y el uso máximo de memoria para cada una. Estos valores se midieron con archivos de entrada de tamaño creciente casi exponencialmente, partiendo con un genoma de 100.000bp hasta uno de alrededor de 11.000.000bp. El rango de tamaño se eligió basándose en lo expuesto sobre el largo de genomas bacterianos en la Sección 2.2.

En la medición del uso de memoria del algoritmo implementado se usaron funciones disponibles en la librería SDSL para obtener el tamaño de todas las estructuras utilizadas y así calcular el consumo máximo de memoria del proceso. La herramienta pilercr imprime en consola el uso máximo de memoria de cada ejecución, se indagó en el código fuente del instrumento y se descubrió que este valor corresponde al máximo de memoria alocada durante la ejecución. Por el otro lado, para medir el uso de memoria de CRT se hizo uso de la herramienta JProfiler. Esta provee un análisis exhaustivo de aplicaciones Java, incluyendo la

evolución del consumo de memoria.

Las pruebas se corrieron en un computador con sistema operativo Ubuntu 18.04 que posee 16Gb de RAM y un procesador Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz. Todos los experimentos se realizaron bajo las mismas condiciones de ejecución.

Los resultados completos de estos experimentos se encuentran en el Apéndice A de este documento.

5.3. Evaluación de Calidad

La precisión y la precisión parcial del algoritmo implementado se encuentran graficadas en la Figura 5.1. Es fácil notar que la precisión de CRISPR detectados completamente es considerablemente baja. La precisión parcial no muestra una tendencia clara, algunos genomas resultaron en una precisión de 100 %, mientras que en otro caso se obtiene un 10 %. El promedio de la precisión del algoritmo es 5 % y 60 % para la precisión parcial. Esto indica que la proporción de verdaderos positivos a falsos positivos es favorable, pero que de todas formas la cantidad de falsos positivos es considerable.

Los genomas con peor precisión acumulada, correspondiente a la suma entre la precisión y la precisión parcial, son NC_013124 y NC_006513. En estos casos se detectaron 8 y 12 falsos positivos respectivamente. La mayoría de estas cadenas son de dos repeticiones, salvo una encontrada en NC_013124 que cuenta con 10 repeticiones.

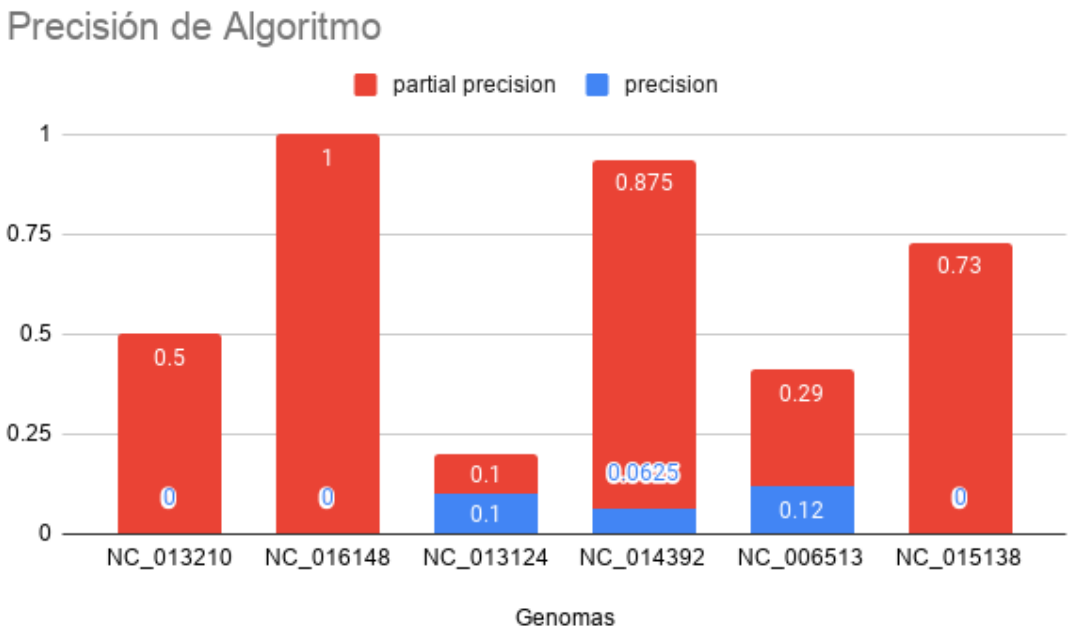


Figura 5.1: Precisión del algoritmo al detectar CRISPR en distintos genomas.

Similarmente, el recall del algoritmo al ser evaluado sobre los genomas de prueba también es bajo, obteniendo 0 en todos los genomas salvo tres. Por el otro lado, el recall parcial es alto para todos los genomas. Al apilar el recall parcial con el recall se obtiene un valor cercano a 100 % en todos los casos. El promedio de recall obtenido es 21 %, mientras que el parcial entrega un promedio de 74 %; sumando estos valores se obtiene un 95 %, por lo que casi todos los CRISPR fueron detectados completa o parcialmente por el algoritmo.

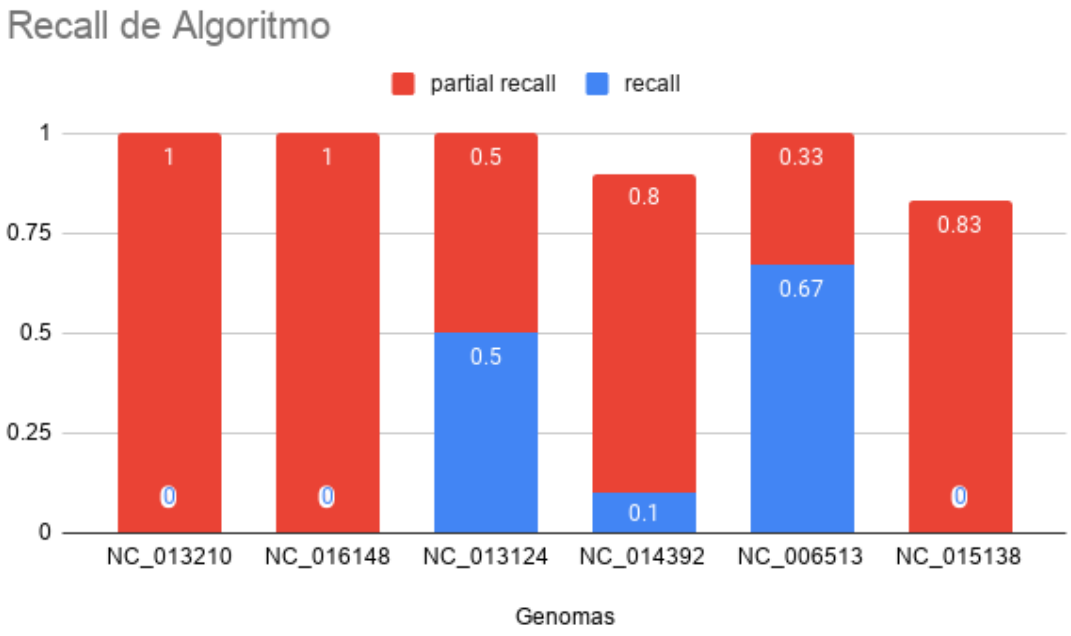


Figura 5.2: Recall del algoritmo al detectar CRISPR en distintos genomas.

Se decidió comparar los resultados obtenidos al buscar CRISPR en el genoma NC_15138 con el algoritmo propuesto, pilercr y CRT. Se eligió este genoma pues presenta variedad en cuanto al largo de cada CRISPR. Un detalle de los CRISPR reportados por cada herramienta se puede encontrar en el Apéndice A. Según CRISPRdb, este genoma cuenta con 6 secuencias CRISPR, no obstante ninguna solución reportó esta cantidad. CRT encontró 4 CRISPR que son un calce perfecto con secuencias presentes en la base de datos, pero falla en detectar dos secuencias compuestas por dos repeticiones. CRT reportó secuencias que calzan con CRISPR estudiados, pero de manera parcial. Del primer CRISPR del genoma solo reportó 19 de sus 48 repeticiones. El algoritmo propuesto encontró 47 de estas ocurrencias, pero de manera fragmentada, es decir se dividió el CRISPR en dos. Solo tres de las secuencias reportadas por el algoritmo no concuerdan con CRISPR de la base de datos, todas las otras forman parte de un CRISPR del genoma. La herramienta pilercr tampoco pudo detectar los CRIPSR más pequeños, mientras que el algoritmo fue capaz de reportar parcialmente uno de estos, omitiendo dos caracteres del DR.

5.4. Evaluación de Uso de Recursos

Los resultados de la evaluación de uso de memoria se encuentran graficados en la Figura 5.3. En el gráfico se puede observar que el algoritmo propuesto presenta un crecimiento lineal y un consumo de memoria menor que las otras herramientas, salvo al actuar sobre el genoma de 11.000.000bp. En este caso el rendimiento de pilercr es mejor que la solución implementada por 10 MB. Mientras que el comportamiento de pilercr es casi constante con una leve alza a partir de las 2.000.000 de bases, CRT fluctúa considerablemente. Esto se ve por ejemplo en la ejecución con 4.296.000bp, tanto el algoritmo implementado como CRT utilizaron un máximo de 39 MB, mientras que al correr los programas con 5 millones de bases CRT utiliza cuatro veces la memoria usada por el algoritmo, alcanzando los 160 MB.

Figura 5.3: Comparación de uso máximo de memoria durante la ejecución de la solución propuesta, pilercr y CRT.

Al observar los resultados de las mediciones de tiempo de ejecución de las distintas soluciones en la Figura 5.4 es clara la gran diferencia entre el algoritmo propuesto y las herramientas existentes. Mientras que estas se mantienen bajo el minuto de ejecución hasta en los genomas más grandes, la solución propuesta supera los 20 minutos al operar sobre el genoma de 11.000.000bp.

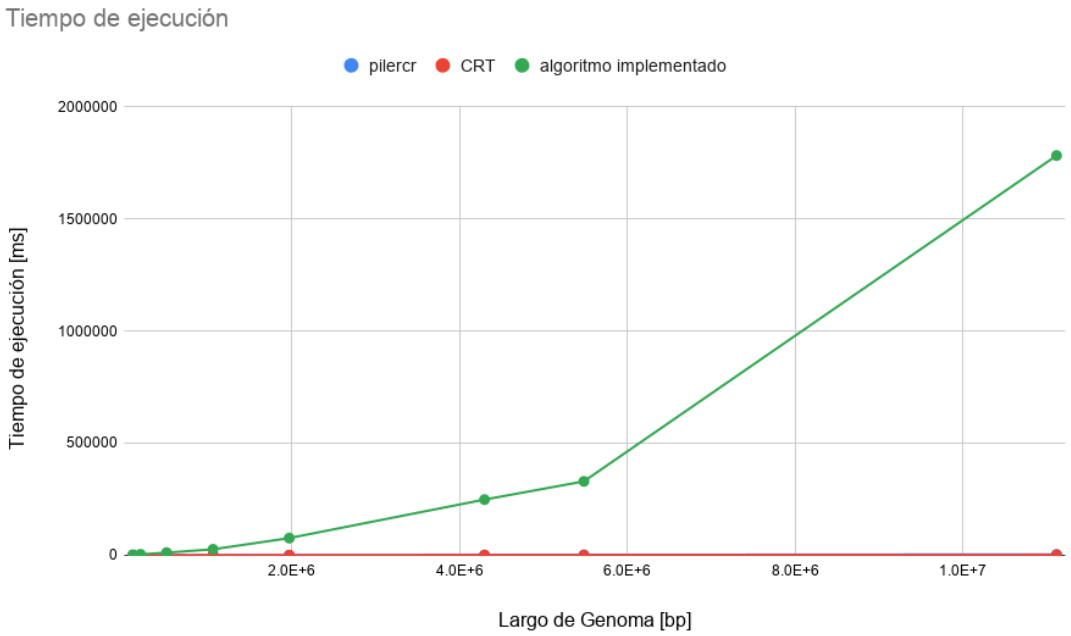


Figura 5.4: Comparación de tiempo de ejecución de algoritmo implementado, pilercr y CRT.

Para apreciar de mejor manera el crecimiento relativo de cada curva se realizó un gráfico aplicando escala logarítmica, que se puede ver en la Figura 5.5. En este se nota una tendencia a crecimiento lineal en las tres soluciones comparadas.

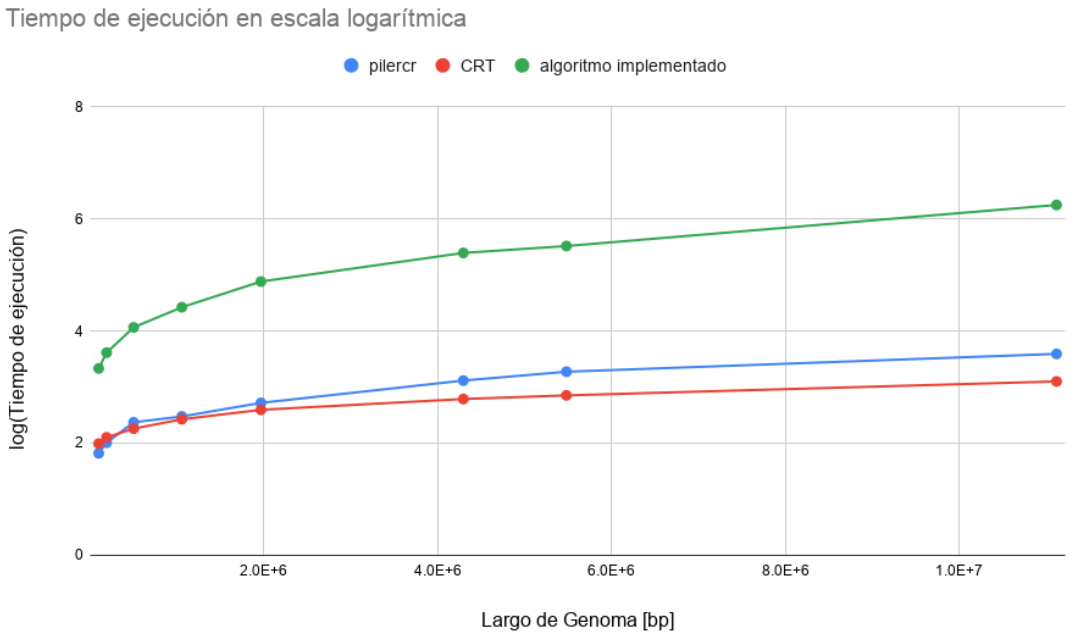


Figura 5.5: Comparación de tiempo de ejecución de algoritmo implementado, pilercr y CRT en escala logarítmica.

Capítulo 6

Aplicación

En este capítulo se compara el comportamiento de las tres soluciones al buscar CRISPR en genomas no estudiados. Se seleccionaron genomas de microbios del Desierto de Atacama, ensamblados a partir de la secuenciación de un metagenoma obtenido del desierto. Se utilizaron dos cepas como datos de entrada, llamadas AC2 y 618, la primera cuenta con 3.810.896 pares de bases nitrogenadas, mientras que la segunda tiene un largo de 4.020.419bp. Los resultados de correr el algoritmo sobre estos genomas se encuentran detallados en el Apéndice A.

Al utilizar las herramientas para detectar CRISPR en la secuenciación de 618 se obtuvo que ni pilercr ni CRT encontraron secuencias CRISPR, mientras que el algoritmo propuesto reportó 4 CRISPR. Las 4 cadenas reportadas por la solución consisten en 2 repeticiones separadas por la distancia necesaria.

En el caso de AC2, las herramientas tuvieron más coincidencias. Si bien todas encontraron un número distinto de CRISPR, las tres concuerdan en cuatro secuencias CRISPR. Sin embargo, el algoritmo propuesto reportó una cantidad significativamente mayor de CRISPR que las otras herramientas, como se ve en la Tabla 6.1.

	CRT	**pilercr**	**algorimo**
618	0	0	4
AC2	5	4	18

Tabla 6.1: CRISPR encontrados por cada herramienta en genomas ensamblados a partir de metagenoma de Desierto de Atacama.

Para ambas cepas los CRISPR encontrados por el algoritmo que no fueron reportados por las otras herramientas corresponden a secuencias de dos repeticiones que cumplen con los requisitos establecidos en este trabajo para ser considerado CRISPR. En el caso de CRT, este reportó una cadena compuesta por repeticiones de largo 19bp. Este valor es inferior al largo mínimo definido en la Sección 2.2, por lo que tiene sentido que las otras herramientas no lo hayan detectado.

Como se mencionó anteriormente, las tres herramientas coincidieron en cuatro secuencias CRISPR del genoma AC2, aunque presentando leves diferencias entre sus reportes. En cuanto al largo y contenido del DR de cada CRISPR encontrado, estos son iguales para todos los instrumentos, mientras que difieren en la cantidad de repeticiones que componen a cada CRISPR. Esto se denota en la Tabla 6.2, en el cual se detalla la cantidad de repeticiones encontradas por cada solución. La columna de posición indica la posición en el genoma de la primera ocurrencia reportada por cualquiera de las soluciones.

CRISPR	**Posición**	**repeticiones CRT**	**repeticiones pilercr**	**repeticiones algoritmo**
AC2_1	1581543	11	8	8
AC2_2	2339046	7	5	6
AC2_3	1340192	6	3	3
AC2_4	2340742	11	10	11

Tabla 6.2: Repeticiones encontradas por cada herramienta en los CRISPR reportados por las tres.

Para todas las cadenas, CRT reporta un número mayor o igual de repeticiones que las otras herramientas, a contrario de pilercr que consistentemente entrega una cantidad menor o igual al resto. Dos casos interesantes son AC2_1 y AC2_3. En este primer CRISPR tanto pilercr como el algoritmo implementado en este trabajo reportan 8 repeticiones, mientras que CRT retornó 11 repeticiones. Al revisar los DR encontrados por cada herramienta, se observó que las 3 repeticiones reportadas únicamente por CRT son diferentes a las otras, la última de estas contando con hasta 10 caracteres distintos al DR del CRISPR. En cuanto a AC2_3, CRT reportó 6 ocurrencias del DR, mientras que pilercr y la solución propuesta encontraron 3 cada una. Las tres repeticiones reportadas por pilercr corresponden a las primeras tres ocurrencias entregadas por CRT, y los DR de estas repeticiones son distintos entre sí. En cambio el algoritmo implementado encuentra las repeticiones que pilercr no encuentra, que sí son idénticas entre sí. La repetición en AC2_2 que fue únicamente encontrada por CRT también presenta varias diferencias con el resto de las repeticiones, un tercio de sus caracteres son distintos al DR de consenso.

Se combinaron los resultados entregados por las tres herramientas para generar un reporte de CRISPR detectados en AC2. Se utilizó como criterio de selección de una repetición haber sido encontrada por al menos dos herramientas o poseer una discrepancia de a lo más 5 % con respecto al DR de consenso. Este último se define como el DR que fue reportado más veces, en caso de repeticiones con alteraciones.

CRISPR	**Posición**	**Largo**	**Largo DR**	**Copias**	**Consenso de DR**
AC2_1	2340742	559	32	9	GTCGCACTCTATATGAGTGCGTGGATTGAAAT
AC2_2	2339113	367	32	6	ATTTCAATCCACGCACTCATGTAGAGTGCGAC
AC2_3	1340192	360	32	6	ATTTCAATCCACGCACTCATGTAGAGTGCGAC
AC2_4	3740063	692	32	11	ATTTCAATCCACGCACTCATGTAGAGTGCGAC

Tabla 6.3: CRISPR detectado en cepa AC2 ensamblada a partir de metagenoma del Desierto de Atacama. Se combinaron los resultados entregados por el algoritmo propuesto, CRT y pilercr.

Capítulo 7

Discusión

Este capítulo está dedicado a la discusión de los resultados obtenidos tras evaluar el rendimiento del algoritmo en cuanto a capacidad de detección y uso de recursos. También se proponen posibles mejoras al algoritmo en base a lo discutido.

7.1. Análisis de Resultados

7.1.1. Pruebas de Calidad

Si bien el algoritmo logra detectar CRISPR en genomas, lo hace manera fragmentada o parcial. Esto quiere decir que entrega secuencias que forman parte de un CRISPR, pero no este en su totalidad o que logra detectar todas las repeticiones de un CRISPR pero con un DR de un largo más corto que el real. Al indagar más en los resultados y en la información dispuesta en CRISPRdb se descubrió que esto se debe a que los CRISPR reales pueden presentar ligeras diferencias entre el DR de una repetición y el DR consensuado del CRISPR.

Si esta alteración se presenta en los extremos del DR, y el substring resultante al eliminar la porción diferente sigue cumpliendo con el requisito de largo mínimo, entonces el algoritmo entrega una cadena CRISPR con un DR más corto que el real pero con el mismo número de repeticiones. Por el otro lado, si un DR contiene una diferencia en su zona central, esto causará que el algoritmo no lo cuente como una repetición pues su texto no coincide con el resto. La consecuencia de esto es que el algoritmo retorna el CRISPR en dos fragmentos: uno conteniendo las repeticiones desde la primera ocurrencia hasta la penúltima antes del DR distinto y otra desde el siguiente del dejado fuera hasta el final. Ambos tipos de alteraciones pueden producirse en un mismo CRISPR más de una vez, resultando en una fragmentación mayor.

Un genoma en el cual ocurren ambos casos es el de *Thermovirga lienii* (NC_016148). Este organismo contiene dos CRISPR, uno de 24 repeticiones y otro de 46. Ambos CRISPR son formados por DR de largo 30. Al ejecutar el algoritmo con este genoma, se obtienen los

resultados expuestos en la Tabla 7.1, en la cual se muestran detalles de cada cadena CRISPR encontrada.

Número de Repeticiones	Largo DR	Posición Primer DR	Posición último DR
24	29	256316	257792
30	30	279012	280930
13	30	281061	281858

Tabla 7.1: Resultados de ejecución sobre NC_016148.

En esta tabla se puede ver que si bien se detectaron las 24 repeticiones del primer CRISPR, el largo DR del CRISPR entregado es 29bp. Al revisar todas las ocurrencias del DR en CRISPRdb se encontró que uno de estos posee una letra alterada con respecto al resto de los DR del CRISPR, esto se puede ver en la Figura 7.1 en la cual se muestra un fragmento de la información sobre el CRISPR en cuestión encontrada en CRISPRdb. Dada la naturaleza del algoritmo, no contabiliza este DR distinto como una ocurrencia. En el proceso de selección de candidatos, al pasar por el nodo equivalente al DR real del CRISPR, el DR alterado no se encontraría en la misma ramificación del árbol de sufijos pues no comparten un prefijo de largo 30. En cambio, el subárbol naciente del substring de largo 29 del DR real sí contiene a todas las ocurrencias reales. Como el algoritmo le da prioridad a las cadenas con más repeticiones, se elige la creada con esas ocurrencias como CRISPR.

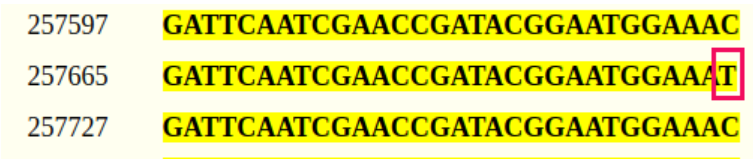


Figura 7.1: Ejemplo de DR alterado en su extremo, en el CRISPR NC_016148_1. Se muestran algunos DR contiguos junto a sus posiciones, y se indica que el DR que parte en 257665 termina con una "T" en vez de una "C".

En la Tabla 7.1 también se indica que el segundo CRISPR del organismo se detectó, pero de forma fragmentada. La primera cadena termina con un DR que se encuentra en la posición 280930 del texto, y la siguiente inicia en la posición 281061. Al buscar estas posiciones en la tabla correspondiente al CRISPR en cuestión en CRISPRdb, se encontró que entre estas dos existe una repetición que cuenta con un DR distinto a las otras, como se ve en la Figura 7.2. Como la alteración del DR se encuentra en la mitad de este y no es posible contar su substring como DR debido a los requisitos de largo, el algoritmo no pudo proceder del mismo modo que en el caso anterior. En esta situación al llevarse a cabo la verificación de candidatos el algoritmo no encontraría una ocurrencia del patrón luego de la posición 280930, por lo que terminaría ahí esa cadena de CRISPR y procedería a comenzar una nueva desde la posición 281061, resultando en dos cadenas.

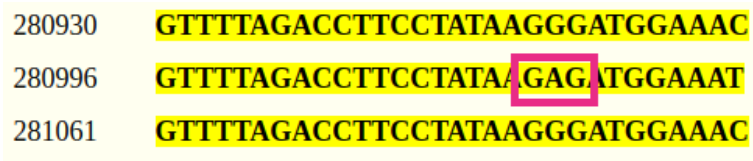


Figura 7.2: Repeticiones de CRISPR_2, el DR que inicia en la posición 280996 presenta dos diferencias al resto de las repeticiones, una de las cuales se encuentra en un punto intermedio del texto.

Este proceso de revisión se llevó a cabo para todas las detecciones parciales y el resultado fue el mismo para cada una de ellas: el motivo por el cual no se detectó el CRISPR completamente es que este contiene algún DR con leves alteraciones. Si estas diferencias no se presentan en el CRISPR, entonces el algoritmo es capaz de detectarlo en su completitud, como se ve en algunos resultados, lamentablemente estos DR con diferencias parecen ser comunes.

De la baja precisión reportada en la Sección 5.3 se puede concluir que existe un alto número de falsos positivos, i.e. que el algoritmo reporta cadenas que no coinciden con CRISPR conocidos. Se verificó manualmente que estas cadenas efectivamente cumplen con los requisitos para ser consideradas CRISPR. En efecto, los DR reportados son todos idénticos dentro de un CRISPR y su largo está en el rango necesario, los spacers que separan las repeticiones también están dentro del rango. Al revisar el contenido del CRISPR reportado en NC_013124 que cuenta con 10 repeticiones, se encontró que todos los spacers que separan cada repetición son iguales. Esto viola una de las características de un CRISPR, que es que sus spacers sean distintos.

También puede ser el caso que estas cadenas categorizadas como falsos positivos en realidad sean CRISPR que no han sido reportados todavía. Esto puede ser el caso ya que, con excepción de la cadena reportada cuyos *spacers* son idénticos, las secuencias reportadas cumplen con todas las características que debe presentar un CRISPR.

Como la cantidad de posibles CRISPR reportados no es tan grande, que la precisión del algoritmo sea relativamente baja no es tan grave ya que verificar los resultados no es costoso. Es preferible que se logre detectar todos los CRISPR aunque se reporten cadenas de más, a que queden secuencias sin reportar. Por esto es más importante que el algoritmo cuente con un recall alto que con una precisión alta.

Al comparar los reportes generados por las tres herramientas evaluadas en este trabajo, se puede notar que el algoritmo detecta secuencias que cumplen con el criterio de CRISPR que las otras no. Esto se puede ver por ejemplo en el caso brevemente estudiado en la Sección 5.3, correspondiente al genoma NC_015138. El algoritmo propuesto detectó 4 CRISPR que las otras herramientas no reportaron, incluyendo uno que se encuentra dentro de la lista de CRISPR conocidos del genoma según CRISPRdb.

Estas omisiones, también vistas al utilizar las herramientas para detectar CRISPR en los genomas de bacterias del Desierto de Atacama, indican que las herramientas están utilizando una heurística de selección que potencialmente puede llevarlos a ignorar CRISPR. La solución

propuesta por el otro lado, no pierde secuencias dado a alguna heurística, sino que por una razón bien definida que corresponde a la alteración presente en algunos DR.

De los resultados obtenidos en estas pruebas también se desprende que en el caso ideal de que un CRISPR no presente ninguna alteración, el algoritmo es capaz de detectarlo y reportarlo en su completitud. Si es que existe una alteración en uno o más de los DR, el CRISPR reporta todo el resto de las ocurrencias, a menos que los DR alterados se encuentren contiguos al DR terminal o inicial.

7.1.2. Pruebas de Rendimiento

La reducción de espacio utilizado es significativa especialmente para los genomas de tamaño menor a 6 millones de pares de bases nitrogenadas. No obstante, el costo temporal del algoritmo es muy elevado hasta para los genomas más pequeños.

Se encapsularon las funciones del algoritmo para medir su tiempo de ejecución por separado, a modo de obtener una noción de cuál etapa del proceso resultaba más costosa en cuanto a tiempo. Al realizar esto se pudo ver que el proceso de filtro de cadenas de CRISPR toma relativamente poco tiempo para los primeros largos de genoma, ya que opera sobre una lista reducida de candidatos. En los genomas más cortos es la selección de candidatos el proceso más costoso, tardando hasta más de diez veces que la verificación de candidatos en algunos genomas.

Por el otro lado, al ejecutar el algoritmo sobre el genoma compuesto por 11 millones de pares de bases nitrogenadas, el costo de verificación aumenta significativamente, convirtiéndose este proceso en el más costoso del algoritmo. Estas mediciones se pueden ver en las Figuras 7.3 y 7.4, en las cuales se compara el tiempo de ejecución de cada etapa.

El elevado costo de la selección de candidatos se debe en parte a que en este proceso se recorre el árbol de sufijos en su completitud, verificando la condición de largo de patrón de cada nodo. En cuanto a la verificación de candidatos seleccionados, la única operación potencialmente costosa es la búsqueda por rango provista por la librería utilizada. Una posible explicación al considerable aumento del costo al trabajar con el genoma más largo es que dicha función no esté basada en la operación *range_next_value* expuesta en la Sección 2.1.2, sino más bien en la operación *range_report* comúnmente usada en árboles wavelet. Esta operación retorna en tiempo $O((1 + occ) \log(\sigma))$ una lista de aquellos puntos de la grilla que se encuentren en determinado rango.

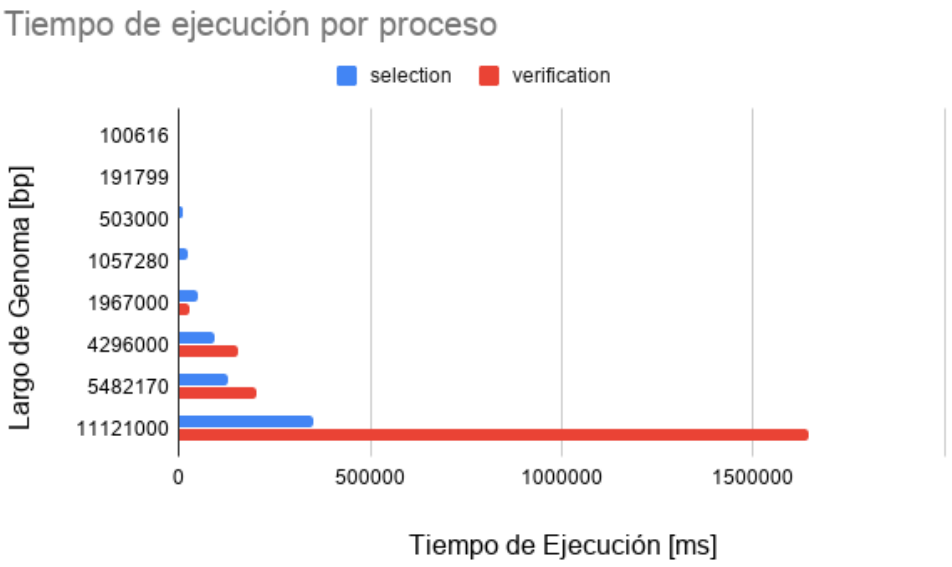


Figura 7.3: Comparación de tiempo de ejecución de etapas principales de algoritmo implementado.

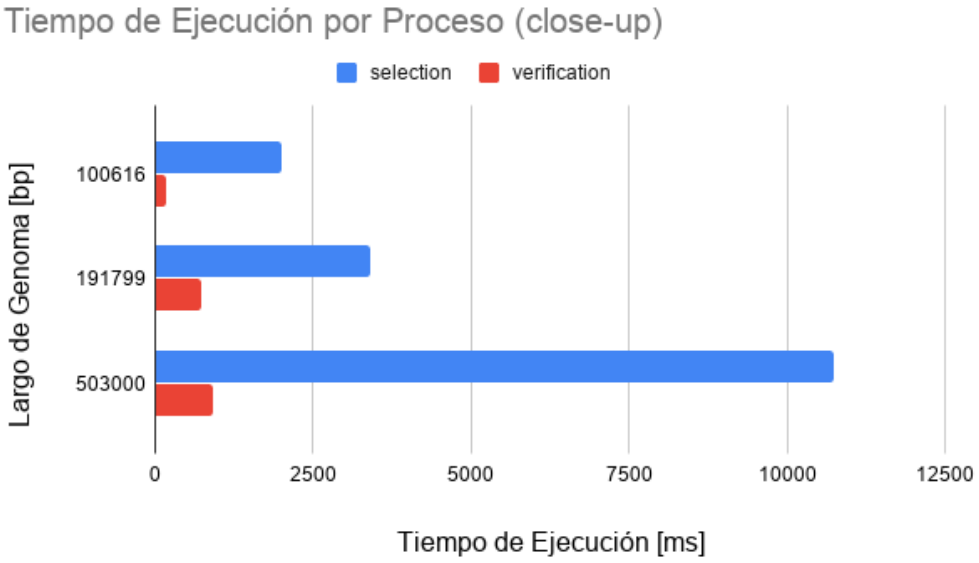


Figura 7.4: Comparación de tiempo de ejecución de etapas principales de algoritmo implementado en genomas más cortos.

7.1.3. Aplicación

El comportamiento del algoritmo propuesto frente a los genomas ensamblados a partir del metagenoma extraído del Desierto de Atacama concuerda con lo observado tras realizar las pruebas de calidad expuestas en la Sección 5.3. La gran cantidad de secuencias reportadas

que no fueron categorizados como CRISPR por las otras herramientas, 18 en total, sugiere que estas utilizan un criterio de selección más ajustado que el propuesto en este trabajo. Esto coincide con la baja precisión obtenida al hacer pruebas.

No obstante, en el caso de detección de CRISPR *de novo*, es más favorable reportar secuencias de más que puedan ser manualmente verificadas en poco tiempo que omitir secuencias que pueden ser relevantes y de utilidad.

Las repeticiones reportadas por las otras herramientas fueron todas igualmente encontradas por el algoritmo implementado, salvo por aquellas que presentan alteraciones en uno o más caracteres de su DR. En el caso de pilercr, el algoritmo pudo encontrar 16 cadenas que esta herramienta no reportó y 5 repeticiones que tanto CRT como el algoritmo reportaron pero pilercr.

7.2. Propuestas de Mejora

7.2.1. Unión de CRISPR Fragmentado

Una primera solución a la fragmentación de cadenas CRISPR que ocurre cuando una repetición presenta una alteración, es intentar unir las cadenas de CRISPR que quedan separadas. Durante el proceso de verificación de candidatos, si no se ha encontrado una ocurrencia en el rango de posiciones posibles y todavía quedan candidatos sin emparejar, se podría buscar si es que la repetición subsiguiente existe dentro de la lista de candidatos. De existir, se podría concatenar a la cadena de ocurrencias. Esto correspondería a "saltarse" una repetición y unir dos fragmentos CRISPR.

Esta medida requeriría que el espacio saltado sea revisado para verificar que efectivamente existe un DR modificado y no se trate de dos CRISPR separados, por lo que se sumaría al costo operacional del algoritmo. También, esta solución solo funcionaría para el caso en que no existen DR alterados contiguos ni que la anomalía ocurra en los extremos del CRISPR.

7.2.2. Reducción de Falsos Positivos

Los resultados de las pruebas realizadas y la aplicación de la herramienta sobre los genomas obtenidos del Desierto de Atacama indican que casi todos los falsos positivos detectados por el algoritmo corresponden a secuencias de dos repeticiones, salvo por una cadena de 10 repeticiones pero cuyos *spacers* no eran únicos. Este último caso no puede ser resuelto sin agregar tiempo computacional pues se tendría que verificar la unicidad de todos los *spacer* de un CRISPR.

Simplemente aumentar el número mínimo de repeticiones que debe tener un CRISPR de dos a tres ciertamente eliminaría todos estos falsos positivos, pero también se perdería la capacidad de encontrar este tipo de CRISPR, que sí se encuentran presente en algunos

organismos como NC_015138. Este *tradeoff* tendría que evaluarse con expertos en CRISPR bacteriano para determinar si sería beneficioso.

Si bien las secuencias consideradas como falsos positivos cumplen con todos los requisitos planteados para ser considerado CRISPR, podría ser el caso que haya una característica de CRISPR que no se esté tomando en cuenta. Por esto se propone evaluar la posibilidad de aumentar las exigencias para que una secuencia sea considerada CRISPR. Se debe indagar nuevamente en los requisitos que usan otras herramientas para seleccionar CRISPR y ver si hay alguno que no se esté tomando en consideración y pueda ser incorporado a la solución entregada en este trabajo.

Asimismo se debe investigar sobre aspectos no considerados respecto a cadenas CRISPR. Una opción podría ser estudiar los genes de proteínas *cas* que acompañan a un CRISPR, mencionados en la Figura 1.1 de este trabajo. De existir un patrón fijo, o un conjunto de estos, que siempre preceden o siguen a un CRISPR, sería posible utilizar esto como un filtro de candidatos y de este modo reducir la cantidad de detecciones incorrectas.

En caso de no existir más características de CRISPR que puedan ser consideradas como requisito por el algoritmo, se propone no cambiar el sistema actual de la herramienta. Si no hay evidencia de que se trata de una detección errónea, podría realmente tratarse de un CRISPR y contener información potencialmente valiosa para investigadores que de no reportarse la secuencia, se perdería. La cantidad de falsos positivos con respecto a CRISPRdb reportados no es suficientemente grande como para imposibilitar la verificación manual en caso de un reporte sospechoso y la ganancia de encontrar un CRISPR de utilidad es muy alta.

7.2.3. Aumento de Rendimiento

Debido a que la etapa más costosa del algoritmo en relación a tiempo de ejecución es la selección de candidatos utilizando un árbol de sufijos, se enfocó en esta para buscar posibles mejoras de rendimiento. La operación fundamental de este proceso es recorrer todos los nodos del árbol de sufijos y revisar si es que la etiqueta del nodo cumple con el requisito de largo mínimo y máximo, y si es que tiene más de un hijo. No es necesario acceder a todos los nodos del árbol de sufijos, se podría buscar la manera de solo iterar dentro de la profundidad deseada, y así disminuir la cantidad de nodos por los cuales se opera.

Una reducción más significativa sería implementar un proceso de filtrado de candidatos que reduzca la cantidad de ocurrencias que deben ser verificadas. Esto podría llevarse a cabo de manera similar al proceso actual de filtrado, eliminando aquellos candidatos cuyas posibles secuencias se encuentran contenidas en las de otros candidatos. De este modo se reduciría significativamente el tiempo de ejecución del algoritmo, pero al posible costo de reducir la calidad de los resultados. Esto podría suceder por principalmente dos razones: el grupo de candidatos que engloba al resto podría no formar un CRISPR mientras que alguno de los eliminados sí, y algún CRISPR formado por este grupo podría no ser maximal, es decir que los candidatos eliminados podrían formar un CRISPR con un DR más largo.

Otro acercamiento que podría ser explorado toma ideas de ambas propuestas. En vez de llevar a cabo el proceso de selección de todos los candidatos y la verificación de estos de forma secuencial, se podría realizar un recorrido por las ramas del árbol y realizar esta verificación por cada rama, deteniendo el descenso por la rama en un momento oportuno. A medida que se avanza por el árbol descendiendo por una rama, si un nodo cumple con las características necesarias para ser seleccionado como candidato, entonces se procede inmediatamente al proceso de verificación de sus hojas.

En esta etapa, si es que la diferencia entre la distancia de una ocurrencia a la siguiente y el largo máximo que puede tener un *spacer* es mayor a la diferencia entre el largo máximo que puede tener una repetición y el largo del candidato sumado, entonces cualquier string del cual el candidato sea prefijo tampoco cumplirá con los requisitos necesarios para formar un CRISPR y la hoja que indica esa posición puede ser eliminada de todo el proceso. Si es que la distancia es menor al valor descrito, entonces se concatenan las ocurrencias posibles y se procede descendiendo por la rama, repitiendo el proceso con los hijos. Si en algún momento al verificar un conjunto de candidatos se corta una cadena de CRISPR que se había podido formar con el padre del candidato, i.e. su predecesor en el árbol de sufijos, se debe retornar la cadena anterior, eliminar las hojas correspondientes del proceso y seguir con las otras cadenas formadas. Que se corte una cadena puede significar que añadir los caracteres que diferencia a un candidato de su predecesor en el árbol de sufijos implica que el texto ya no forme parte de un CRISPR, o que existe una repetición que posee una alteración, como fue explicado en la Sección 7.1.1. En el momento en que se hayan descartado todas las hojas de un nodo, ya sea porque se formaron las cadenas de CRISPR maximales o porque no cumplían con los requisitos para ser CRISPR, se detiene el descenso por la rama y se devuelven los resultados encontrados.

Este método podría permitir reutilizar información y eliminar operaciones redundantes, por lo que pareciera ser una buena ruta exploratoria a seguir para mejorar el trabajo presentado en este documento.

En cuanto a la reducción del tiempo de ejecución de la verificación de candidatos, se propone en primer lugar realizar una investigación exhaustiva acerca del método utilizado por la librería en su función de búsqueda por rango. Si es que lo encontrado no es el todo satisfactorio, se propone implementar la operación *range_next_value* y probar si mejora el rendimiento del algoritmo al reemplazar el método de búsqueda por esta nueva función.

Capítulo 8

Conclusión

La detección de CRISPR *de novo* en metagenomas permite obtener un mayor entendimiento sobre la interacción entre microbios y su ambiente. En particular en el Desierto de Atacama estudiar estas interacciones es de interés dado a la caracterización de los microbios que residen en este ambiente como extremófilos. El análisis genómico tiende a presentar altos costos en cuanto a uso de recursos, debido al gran tamaño que puede tener una secuencia de ADN. En torno a este contexto, el trabajo realizado durante esta memoria consistió en diseñar e implementar un algoritmo detector de CRISPR que haga uso de estructuras de datos compactas, con el fin de determinar si el uso de este tipo de estructuras permite mejorar el rendimiento de herramientas detectoras de CRISPR.

El proyecto tuvo como hito crítico el descubrimiento de que un CRISPR no tiene un número mínimo de repeticiones. La estrategia inicial planteaba utilizar este requisito como método de filtro y así mantener un bajo tiempo de ejecución. Este descubrimiento llevó a un replanteamiento del sistema de filtro y verificación de candidatos, que desembocó en la solución presentada en este trabajo.

En retrospectiva, se debió haber estudiado a mayor profundidad al respecto de CRISPR y consultado con aun más expertos al respecto, pues se fueron aprendiendo nuevas cosas acerca de este sistema a medida que se avanzaba en el trabajo. Esto demuestra que los estudios previos y las consultas realizadas no fueron suficientemente exhaustivas, lo que se tendrá en consideración en trabajos futuros.

Se encontraron dificultades al momento de implementar la solución, principalmente por la complejidad que presenta trabajar con estructuras de datos compactas pues a pesar de contar con una librería que abstrae distintas funcionalidades, se opera con estructuras no comunmente usadas como bitmaps y secuencias de paréntesis balanceados.

Un desafío encontrado durante este trabajo fue la validación de la herramienta en cuanto a la calidad de los resultados entregados. Esto se debe a la fragmentación que ocurre al reportar cadenas de CRISPR que contienen un DR con una alteración. Al principio no se sabía la causa de esta fragmentación y se creía erróneamente que era a causa de un error del algoritmo, por lo que se utilizó mucho tiempo buscando la fuente del error en el código.

Luego de encontrar la verdadera razón, se tuvo que adaptar las pruebas para contabilizar detecciones parciales automáticamente.

Según lo descrito a lo largo del documento, se puede concluir que los objetivos específicos propuestos se han cumplido. Se diseñó e implementó una estrategia que utiliza árboles de sufijos compacto y árboles wavelet para encontrar CRISPR en genomas bacterianos. Una vez desarrollada esta solución, se evaluó la calidad de los resultados producidos al buscar CRISPR en genomas ya estudiados. Asimismo se midió el rendimiento del algoritmo y herramientas existentes en cuanto a uso de recursos, comparándolos entre sí. Finalmente se aplicaron los programas sobre genomas ensamblados a partir del metagenoma extraído del Desierto de Atacama, produciendo así un listado de los CRISPR encontrados en ellos.

El algoritmo implementado logró detectar casi todos los CRISPR del conjunto de pruebas, obteniendo un promedio de recall total de 95 %. Sin embargo, también reportó muchas secuencias que no están categorizadas como CRISPR en CRISPRdb, a pesar de cumplir con los requisitos necesarios para ser un CRISPR. Por esta razón se obtuvo un promedio de precisión de 65 %.

En cuanto al consumo máximo de memoria, el algoritmo presenta una ganancia en este aspecto en comparación con las otras herramientas. Los resultados obtenidos indican un crecimiento lineal al tamaño del genoma y que su uso máximo de memoria es menor al de las soluciones existentes para genomas de tamaño menor a 8.000.000 pares de bases nitrogenadas. En lo que respecta el tiempo de ejecución, el algoritmo tuvo un rendimiento altamente desfavorable en comparación a las otras herramientas, tardando casi 100 veces más en encontrar CRISPR hasta en el genoma de prueba más pequeño.

Fue posible encontrar 4 secuencias CRISPR en el genoma de una de las cepas ensambladas a partir del metagenmas obtenido del Desierto de Atacama, al unificar los resultados entregados por las tres herramientas.

Dado lo anterior, el objetivo general se ha cumplido. Fue posible determinar si el rendimiento de algoritmos detectores de CRISPR *de novo* puede mejorar con el uso de estructuras de datos compactas. Si bien el espacio utilizado por el algoritmo es significativamente menor al de las otras herramientas, el *tradeoff* entre el uso de memoria y tiempo de ejecución hace que la solución propuesta no sea viable en cuanto a uso de recursos. A la vez, los resultados retornados son complementarios a los de pilercr, y un poco redundantes a los obtenidos al ejecutar CRT. La herramienta es capaz de reportar secuencias cortas que presentan características de CRISPR, mientras que los otros instrumentos omiten este tipo de secuencias.

En este trabajo se logró diseñar e implementar una estrategia que garantiza reportar todos los patrones repetidos e interespaciados según la definición de CRISPR utilizada. Los resultados indican que sí es posible detectar CRISPR *de novo* en genomas de bacterias utilizando estructuras de datos compactas, por lo que se plantea este trabajo como una base de partida de la cual se puede seguir explorando la temática.

En este documento también se proponen algunos acercamientos e indicaciones que se pueden estudiar para avanzar el trabajo, mejorando la calidad de los resultados y el tiempo de ejecución. Un aspecto fundamental en el que se propone trabajar es el de permitir que

existan leves diferencias entre los DR de un CRISPR, ajustando más así la solución a la realidad. Este camino implicaría replantear la solución propuesta en este trabajo al requerir el procesamiento de más de un rango lexicogŕafico a la vez.

Bibliografía

[1] Bland, Charles, Teresa L. Ramsey, Fareedah Sabree, Micheal Lowe, Kyndall Brown, Nikos C. Kyrpides y Philip Hugenholtz: *CRISPR Recognition Tool (CRT): a tool for automatic detection of clustered regularly interspaced palindromic repeats*. BMC Bioinformatics, 8(1):209, Jun 2007.

[2] diCenzo, George C. y Turlough M. Finan: *The Divided Bacterial Genome: Structure, Function, and Evolution*. Microbiology and Molecular Biology Reviews, 81(3):e00019–17, 2017, ISSN 1092-2172. `https://mmbr.asm.org/content/81/3/e00019-17`.

[3] Edgar, Robert C.: *PILER-CR: Fast and accurate identification of CRISPR repeats*. BMC Bioinformatics, 8(1):18, Jan 2007, ISSN 1471-2105. `https://doi.org/10.1186/1471-2105-8-18`.

[4] Gagie, Travis, Gonzalo Navarro y Simon J. Puglisi: *New algorithms on wavelet trees and applications to information retrieval*. Theoretical Computer Science, 426-427:25 – 41, 2012, ISSN 0304-3975. `http://www.sciencedirect.com/science/article/pii/S0304397511009625`.

[5] Gawrychowski, Pawel, Tomohiro I, Shunsuke Inenaga, Dominik Köppl y Florin Manea: *Efficiently Finding All Maximal α-gapped Repeats*. CoRR, abs/1509.09237, Septiembre 2015. `http://arxiv.org/abs/1509.09237`.

[6] Gog, Simon, Timo Beller, Alistair Moffat y Matthias Petri: *From Theory to Practice: Plug and Play with Succinct Data Structures*. En *13th International Symposium on Experimental Algorithms, (SEA 2014)*, páginas 326–337, 2014.

[7] Grissa, I., G. Vergnaud y C. Pourcel: *The CRISPRdb database and tools to display CRISPRs and to generate dictionaries of spacers and repeats*. BMC Bioinformatics, 8:172, May 2007.

[8] Haft, Daniel, Jeremy Selengut, Emmanuel Mongodin y Karen Nelson: *A guild of 45 CRISPR-associated (Cas) protein families and multiple CRISPR/Cas subtypes exist in prokaryotic genomes*. PLoS computational biology, 1:e60, Diciembre 2005.

[9] Jinek, Martin, Krzysztof Chylinski, Ines Fonfara, Michael Hauer, Jennifer A. Doudna y Emmanuelle Charpentier: *A Programmable Dual-RNA–Guided DNA Endonuclease in Adaptive Bacterial Immunity*. Science, 337(6096):816–821, 2012, ISSN 0036-8075.

https://science.sciencemag.org/content/337/6096/816.

[10] Kunin, Victor, Rotem Sorek y Philip Hugenholtz: *Evolutionary conservation of sequence and secondary structures in CRISPR repeats*. Genome Biology, 8(4):R61, 2007. https://doi.org/10.1186/gb-2007-8-4-r61.

[11] Lei, Jikai y Yanni Sun: *Assemble CRISPRs from metagenomic sequencing data*. Bioinformatics, 32(17):i520–i528, Agosto 2016, ISSN 1367-4803. https://doi.org/10.1093/bioinformatics/btw456.

[12] Navarro, Gonzalo: *Wavelet Trees for All*. En Kärkkäinen, Juha y Jens Stoye (editores): *Combinatorial Pattern Matching*, páginas 2–26, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg, ISBN 978-3-642-31265-6.

[13] Pourcel, Christine, Gilles Vergnaud y Ibtissem Grissa: *CRISPRFinder: a web tool to identify clustered regularly interspaced short palindromic repeats*. Nucleic Acids Research, 35(suppl_2):W52–W57, Julio 2007, ISSN 0305-1048.

[14] Pride, David T., Julia Salzman y David A. Relman: *Comparisons of clustered regularly interspaced short palindromic repeats and viromes in human saliva reveal bacterial adaptations to salivary viruses*. Environmental Microbiology, 14(9):2564–2576, 2012. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1462-2920.2012.02775.x.

[15] Ruiquan, Ge, Guoqin Mai, Pu Wang, Manli Zhou, Youxi Luo, Yunpeng Cai y Fengfeng Zhou: *CRISPRdigger: detecting CRISPRs with better direct repeat annotations*. Scientific Reports, 6:32942, Septiembre 2016.

[16] Sadakane, K.: *Compressed Suffix Trees with Full Functionality*. Theory of Computing Systems, 41(4):589–607, 2007.

[17] Skennerton, Connor T., Michael Imelfort y Gene W. Tyson: *Crass: identification and reconstruction of CRISPR from unassembled metagenomic data*. Nucleic Acids Research, 41(10):e105–e105, Marzo 2013, ISSN 0305-1048. https://doi.org/10.1093/nar/gkt183.

[18] Terns, Michael P y Rebecca M Terns: *CRISPR-based adaptive immune systems*. Current Opinion in Microbiology, 14(3):321 – 327, 2011, ISSN 1369-5274. http://www.sciencedirect.com/science/article/pii/S1369527411000488, Ecology and industrial microbiology / Special section: Archaea.

[19] Ussery, Dave y Peter Hallin: *Genome Update: length distributions of sequenced prokaryotic genomes*. Microbiology (Reading, England), 150:513–6, Abril 2004.

Apéndice A

Resultados

CRISPR	DR	largo DR	repeticiones	posición inicial
NC_013210_1	GTGTTCCCCGCACACGCGGGGATGAACCG	29	24	123524
NC_016148_1	GATTCAATCGAACCGATACGGAATGGAAAC	30	24	256316
NC_016148_2	GTTTTAGACCTTCCTATAAGGGATGGAAAC	30	46	279012
NC_013124_2	GGATCACCCCCGCCTGCGCGGGGAGCAC	28	28	996601
NC_013124_3	GGATCACCCCCGCCTGCGCGGGGAGCAC	28	41	1000880
NC_014392_1	GTTTTATCTGAACTATGAGGGATGTAAAC	29	178	145343
NC_014392_2	GTTTTATCTGAACTATGAGGGATGTAAAC	29	15	157108
NC_014392_3	GTTTTATCTGAACTATGAGGGATGTAAAC	29	4	160079
NC_014392_4	GTTTGTAGCCTTCCATTTGGGGATTGAAAC	30	17	2433976
NC_014392_5	GTTTGTAGCCTCCCCTTTGGGGATTGAAAC	30	2	2443408
NC_014392_6	GTTTGTAGCCTCCCCTTTGGGGATTGAAAC	30	24	2449154
NC_014392_7	GTTTGTAGCCTCCCCTTTGGGGATTGAAAC	30	23	2460383
NC_014392_8	GTTTGTAGCCTCCCCTTTGGGGATTGAAAC	30	30	2464720
NC_014392_9	GTTTGTAGCCTCCCCTTTGGGGATTGAAAC	30	20	2477566
NC_014392_10	GTTTGTAGCCTCCCCTTTGGGGATTGAAAC	30	7	2486476
Nc_006513_1	GTTTCAATCCACGCCCCCCGTCACCGAGGGGCGATGC	37	76	1941465
Nc_006513_2	GTTTCAATCCACGCCCCCCGTCACCGAGGGGCGATGC	37	28	1948501
Nc_006513_3	GTTTCAATCCACGCCCCCCGTCACCGAGGGGCGATGC	37	14	1952604
NC_015138_1	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC	36	48	300343
NC_015138_2	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC	36	2	304387
NC_015138_3	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC	36	13	305395
NC_015138_5	GTTTCAACCCACGCGCCCGCACAGGGCGCGAC	32	2	1785060
NC_015138_6	GTTTCAACCCACGCGCCCGCACAGGGCGCGAC	32	21	1785247
NC_015138_7	GTCGCGCCCCGCGTGGGCGCGTGGGTTGAAAC	32	52	4426774

Tabla A.1: CRISPR contenido en genomas participantes en prueba de calidad.

Copies	DR length	Position of first DR	Position of last DR	DR
2	26	34275	34331	AGGACAGAAGGACAGAAGGACAGAAG
24	28	123524	124927	GTGTTCCCCGCACACGCGGGGATGAACC

Tabla A.2: CRISPR reportados por algoritmo en NC_013210.

Copies	DR length	Position of first DR	Position of last DR	DR
24	29	256316	257792	GATTCAATCGAACCGATACGGAATGGAAA
30	30	279012	280930	GTTTTAGACCTTCCTATAAGGGATGGAAAC
13	30	281061	281858	GTTTTAGACCTTCCTATAAGGGATGGAAAC

Tabla A.3: CRISPR reportados por algoritmo en NC_016148.

Copies	DR length	Position of first DR	Position of last DR	DR
2	29	444592	444656	GGCACCATCGGTGCGGTGTGCGATCGGGT
2	26	723786	723840	CCGGCGCCACAGCCCGCGTCGGCGCC
27	28	996662	998249	GGATCACCCCCGCCTGCGCGGGGAGCAC
41	28	1000880	1003321	GGATCACCCCCGCCTGCGCGGGGAGCAC
2	36	1390443	1390507	ATTGTTCCCAGCGGATATTGTTCCCAGCGGATATTG
2	32	1460406	1460466	TGATCCTCTTAGAGTTGATCCTCTTAGAGTTG
10	34	1516994	1517606	GGCTGGTGTGCCTAGGAGGGGATGGGTGCCTCCC
2	26	1570415	1570469	CCGGCAACGCCAGCCACTCCGGCGAC
2	32	1572125	1572185	ACACGACATCCACGTCGCAGGACCAGTCGAGC
2	30	1742184	1742245	GGGGCATCTACGTGCGCGCGAGGCGCGTCG

Tabla A.4: CRISPR reportados por algoritmo en NC_013124.

Copies	DR length	Position of first DR	Position of last DR	DR
178	29	145343	156999	GTTTTATCTGAACTATGAGGGATGTAAAC
2	29	157041	157108	GTTTTATCTGAACTATGAGGGATGTAAAC
8	27	157241	157701	GTTTTATCTGAACTATGAGGGATGTAA
2	27	157768	157834	GTTTTATCTGAACTATGAGGGATGTAA
2	30	157900	157966	AGTTTTATCTGAACTATGAGGGATGTAAAC
3	29	160079	160211	AGTTTTATCTGAACTATGAGGGATGTAAAC
2	31	2433976	2434043	GTTTTATCTGAACTATGAGGGATGTAAAC
12	30	2434114	2434844	GTTTGTAGCCTTCCATTTGGGGATTGAAACG
22	30	2449154	2450550	GTTTGTAGCCTTCCATTTGGGGATTGAAAC
19	30	2460383	2461576	GTTTGTAGCCTCCCCTTTGGGGATTGAAAC
16	30	2464720	2465723	GTTTGTAGCCTCCCCTTTGGGGATTGAAAC
12	30	2465855	2466586	GTTTGTAGCCTCCCCTTTGGGGATTGAAAC
2	25	2466586	2466650	GTTTGTAGCCTCCCCTTTGGGGATT
19	30	2477566	2478762	GTTTGTAGCCTCCCCTTTGGGGATTGAAAC
5	30	2486542	2486810	GTTTGTAGCCTCCCCTTTGGGGATTGAAAC

Tabla A.5: CRISPR reportados por algoritmo en NC_14392.

Copies	DR length	Position of first DR	Position of last DR	DR
2	26	109428	109482	GAGCGCCCGGAAAAGGTTGAGCGCCC
2	42	914979	915052	CGAAGACAAAACCAAAGGACAAAACCAAAGGACAAAACCAAA
2	27	936021	936081	GGATCAGTCGGACCGGGATCAGTCGGA
2	24	1047245	1047302	GCCGTCACTTCGACCGTCTGCGTG
3	29	1047311	1047425	TCGACCGTCTGCGTGCCGGTCGTCACCGT
2	23	1047502	1047559	CGTCACCGTCTCGGTCGCCGCTT
2	24	1048411	1048468	GGTCGCCGTCAGCTCGATCGTCTG
2	29	1048478	1048535	AGCTCGATCGTCTGCGTGCCCGTCGTCAC
2	25	1048613	1048670	GTCGTCACCGTTTCGGTCGCAGCTT
2	41	1294422	1294491	CCCTGACCGGGCTGCTTCTGCGTACCCTGGCTTTGCTCCTG
5	37	1941537	1941827	GTTTCAATCCACGCCCCCCGTCACCGAGGGGCGATGC
8	37	1941902	1942405	GTTTCAATCCACGCCCCCCGTCACCGAGGGGCGATGC
21	37	1942481	1943920	GTTTCAATCCACGCCCCCCGTCACCGAGGGGCGATGC
34	37	1944001	1946375	GTTTCAATCCACGCCCCCCGTCACCGAGGGGCGATGC
7	37	1946451	1946881	GTTTCAATCCACGCCCCCCGTCACCGAGGGGCGATGC
28	37	1948501	1950432	GTTTCAATCCACGCCCCCCGTCACCGAGGGGCGATGC
14	37	1952604	1953539	GTTTCAATCCACGCCCCCCGTCACCGAGGGGCGATGC
2	24	4228421	4228481	TGTCCACCTCACTGTCCACCTCAC
2	47	4266042	4266126	TCGGTGATGGGGCGCTGCCGGGTCATGCGGTAGCGGGCGAGGTCGGT

Tabla A.6: CRISPR reportados por algoritmo en NC_006513.

Copies	DR length	Position of first DR	Position of last DR	DR
40	36	300343	302918	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
7	36	303050	303445	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
2	34	304389	304455	TCTAGATCACTGGGATATGCGCACTGGCCGGAAC
10	36	305395	305989	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
2	47	871371	871455	TTGGCAGGAGCTGCGGCCTTCTTTGCCGGTGCAGCTGCCTTCTTGGC
18	32	1785448	1786583	GTTTCAACCCACGCGCCCGCACAGGGCGCGAC
2	47	2271473	2271554	GATGTCCAGGCCCCAGGTCGGCTGGTGGGCGACGATGAGCCGCGGAG
2	47	3544477	3544561	GCCCCCACGCTCCCGGCTTCGCCTGGTTCGCTGCCCCCCGGGGGGGC
29	32	4426774	4428643	GTCGCGCCCCGCGTGGGCGCGTGGGTTGAAAC
5	32	4428774	4429038	GTCGCGCCCCGCGTGGGCGCGTGGGTTGAAAC
13	32	4429181	4429983	GTCGCGCCCCGCGTGGGCGCGTGGGTTGAAAC

Tabla A.7: CRISPR reportados por algoritmo en NC_015138.

Copies	DR length	Position of first DR	Position of last DR	DR consensus
19	36	302259	303445	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
12	36	305395	306121	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
18	32	1785448	1786583	GTTTCAACCCACGCGCCCGCACAGGGCGCGAC
34	32	4426774	4428972	GTCGCGCCCCGCGTGGGCGCGTGGGTTGAAAC
16	32	4429181	4430183	GTCGCGCCCCGCGTGGGCGCGTGGGTTGAAAC

Tabla A.8: CRISPR reportaos por pilercr en NC_015138.

copies	DR length	Position of first DR	Position of last DR	DR consensus
48	36	300343	303445	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
13	36	305395	306187	AGTCTAGATCACTGGGATATGCGCACTGGCCGGAAC
21	32	1785247	1786583	GTTTCAACCCACGCGCCCGCACAGGGCGCGAC
52	32	4426774	4430183	GTCGCGCCCCGCGTGGGCGCGTGGGTTGAAAC

Tabla A.9: CRISPR reportados por CRT en NC_015138.

largo	algoritmo	pilercr	CRT
100616	0.54	39	18.12
191799	1.07	39	18.08
503000	2.87	40	44.97
1057280	6.71	41	108
1967000	12.32	42	57.9
4296000	29.48	48	29.34
5482170	37.58	51	157.8
11121000	79.48	67	130

Tabla A.10: Resultados de prueba de uso máximo de memoria por cada herramienta. El consumo máximo se muestra en megabytes.

largo	algoritmo	pilercr	CRT
100616	5630	66	98
191799	11756	102	127
503000	18355	236	182
1057280	39397	300	267
1967000	101570	524	394
4296000	350977	1311	614
5482170	428684	1885	713
11121000	1354968	3924	1265

Tabla A.11: Resultados de pruebas de tiempo de ejecución, medido en milisegundos.

Copies	DR length	Position of first	DR Position of last DR	DR
2	29	281115	281175	GGCTGAGGTTCGCCCCCTCCATGCGCGCC
2	29	1407299	1407365	GCGAGGTCGGTGACGCCGGTTTGGTCGAG
2	47	1435329	1435308	CGGGGCCGCACCGGGGGCCGGGGTGATCGCCGGGGTGGGCCGGGTGC
2	47	1624294	1624371	GATTGCCCCAGAGCGTGAGCAGCGTGCCAAGGACCATAAATACCACC

Tabla A.12: CRISPR encontrados por el algoritmo en cepa 618.

Copies	DR length	Position of first DR	Position of last DR	DR
2	47	489515	489593	GGCGCAACAGGGACAACTGGAGCCACGGGTTCAACAGGAGTAACCGG
2	23	489758	489812	GCCACAGGTTCAACAGGAGTAAC
2	47	969150	969230	TCCTCCCTTAACTTTAATAATTTGTTAGTTTTGTACCTGGATAGTAG
8	32	1581508	1581969	GTCGCACTCTATATGAGTGCGTGGATTGAAAT
6	32	2339153	2339488	ATTTCAATCCACGCACTCATGTAGAGTGCGAC
2	34	2340005	2340071	ATTTCAATCCACGCACTCTTGTAGAGTGCGACTG
3	33	2340432	2340562	ATTTCAATCCACGCACTCATGTAGAGTGCGACT
11	32	2340685	2341345	ATTTCAATCCACGCACTCATGTAGAGTGCGAC
2	47	2444463	2444544	CCTGTATCTCCCGTTGCACCTGTGGCTCCCGTTACTCCGGTCGCTCC
2	47	3505315	3505395	CATTCGGTCAATATGCGATAGAAGCAATTGGCTGGGTATGGAACAAA
3	26	3739852	3739960	GCTTCTTCGGCCAACCTGGCTTCTTC
2	47	3741385	3741466	TCGAATACTGTTTTTATCAGCCCGGTCATTTCGGTATCTGATGTATG
2	27	3776117	3776172	GCCCGGCCAATTTTTTCGGCCACGTGC
2	29	3787044	3787101	GAAGAACCATCCTCCCCGAAATCAATAAT
2	26	3797561	3797615	GCTTCTTCGGCCAACCTGGCTTCTTC

Tabla A.13: CRISPR reportados por algoritmo en cepa AC2 del Desierto de Atacama.