Universidad de Concepción

UNIVERSIDAD DE CONCEPCIÓN

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

CRITERIOS DE SELECCIÓN BASADOS EN SKETCHES PARA EL PROBLEMA DE SIMILITUD GENÓMICA

Tesis presentada para optar al grado de Magister en Ciencias de la Computación

Autor: Álvaro Guzmán Chacón

Supervisores:

Cecilia Hernández Rivas

Miguel Figueroa Toro

Índice general

Índice de figuras

Índice de cuadros

Resumen

El cómputo de similitud genómica es importante para distintas aplicaciones como clustering y clasificación taxonómica en área de genómica y metagenómica. Una de las métricas de similitud más utilizadas es el coeficiente de Jaccard.

Debido al alto costo computacional de métodos tradicionales, la comunidad científica ha propuesto métodos de estimación basados en estructuras de datos probabilísticas llamadas sketches. Sin embargo, estos métodos aún pueden requerir un alto tiempo de cómputo para resolver el problema de encontrar todos los pares de secuencias genómicas cuya similitud supere un umbral. Una alternativa para enfrentar esta dificultad es utilizar algoritmos que permitan decidir, con alta probabilidad, si un par no es suficientemente similar, permitiendo reducir el tiempo de cómputo. A este tipo de algoritmos es a al que llamamos criterio de selección, ya que clasifica pares entre similares y no similares.

Este trabajo propone dos criterios de selección basados en propiedades de la confianza en la estimación de sketches de cardinalidad y uno basado en el sketch SuperMinHash con LSH. Los criterios son implementados y validados usando la colección de genomas RefSeq. Los resultados obtenidos muestran una alta sensibilidad y especificidad de los criterios propuestos, mejorando el tiempo de cómputo entre 2 y 10 veces usando cerca del 1 % de espacio adicional.

Capítulo 1

Introducción

Comparar la similitud entre conjuntos de datos de ADN es una herramienta muy importante en el campo de la biología, con aplicaciones en el estudio de la relación entre bacterias [12] y, en general, en estudios de organismos biológicos [11]. Una medida muy utilizada para establecer similitud entre secuencias de ADN es la identidad nucleótida media (ANI por sus siglas en inglés) [15]. Por lo general, un ANI de un 95 % entre secuencias indica una relación fuerte.

Computar el ANI requiere de alineamiento de secuencias, lo cual es muy costoso computacionalmente. Además, las tecnologías modernas de secuenciación masiva han permitido la generación de colecciones grandes de bases de datos genómicas que requieren de algoritmos escalables que permitan su procesamiento [24, 26, 27]. Por este motivo surge la necesidad de desarrollar nuevas técnicas capaces de realizar estimaciones de medidas de similitud y que puedan manejar la gran escala de cantidad de datos actual.

En este contexto aparecen las estructuras probabilistas llamadas **sketches**, estas corresponden a estructuras de datos que ocupan espacio limitado (típicamente sublineal) que permiten estimar alguna función de interés de algún conjunto de datos. Por ejemplo, MinHash [4] y SuperMinHash [8] son sketches que estiman la similitud entre conjuntos. Por otro lado, HyperLogLog [14] es un sketch que estima la cardina-

lidad de conjuntos. Gracias a estas estructuras es que aparecen herramientas como MASH [21] y Dashing [1]. A pesar de utilizar sketches distintos, ambas herramientas comparten el objetivo de estimar similitud genómica.

Sin embargo, cuando se tiene una gran base de datos, una herramienta como Dashing calcula la similitud genómica entre todos los pares posibles de secuencias y, en general, uno necesita sólo pares de secuencias que cumplan con una similitud mínima, como el ANI del 95 % mencionado al inicio. Esto lleva a la pregunta, ¿es posible reducir el número de comparaciones en base a las condiciones sobre similitud?

El índice de Jaccard [23] es un índice de similitud entre dos conjuntos que toma valores entre 0 y 1, donde 0 indica conjuntos totalmente diferentes y 1, conjuntos iguales. Este coeficiente se denota por $J(A, B)$, donde A y B son los conjuntos a comparar. Dado que una secuencia de ADN se puede representar como un conjunto de sub-secuencias de largo k, llamado conjunto de k-mers [19], se puede estimar la similitud entre dos secuencias a través del correspondiente índice de Jaccard entre los conjuntos de k-mers. Este índice es muy relevante en el estudio de similitud genómica, pues presenta una fuerte correlación con el ANI y la distancia MASH [15, 21]. Así, la desigualdad $h \leq J(A, B)$, con h entre 0 y 1, es una manera de exigir que los pares de conjuntos A y B sean suficientemente similares. Luego, si se tienen N secuencias, comparar a todos los pares resulta en $N(N-1)/2$ comparaciones, es decir, es $\mathcal{O}(N^2)$ con respecto a la cantidad de secuencias. Dado que las bases de datos crecen rápidamente, es necesario encontrar métodos que permitan agilizar al proceso de comparación.

Un criterio de selección es un algoritmo de decisión que permite clasificar pares como similares o no similares, con el objetivo de tener una sensibilidad de 1, y una especificidad también lo más cercana a 1 posible. Además será necesario que un criterio de selección se calcule rápido en comparación a calcular el coeficiente de Jaccard, pues en caso contrario será más rápido y eficiente calcular el mismo coeficiente de Jaccard.

Buscar sólo los pares que cumplan con tener un coeficiente de Jaccard mayor a h permite definir un criterio de selección determinista dado por las cardinalidades de

los conjuntos en cuestión [24]. Este criterio logra reducir el número de comparaciones entre conjuntos de k-mers que se realizan, al costo mínimo de comparar sólo las cardinalidades individuales de ambos conjuntos, acelerando así el proceso de selección. El objetivo del presente estudio es analizar este criterio de selección directo y definir nuevos criterios que permitan reducir el número de comparaciones a través de una comparación entre los conjuntos de k-mers, y así reducir el tiempo de cómputo.

El desarrollo de esta investigación se presenta en este documento a través de varios capítulos. El capítulo **Marco teórico** presenta las bases y conceptos previos necesarios para comprender esta investigación. El siguiente capítulo, **Estado del arte**, documenta distintas investigaciones y trabajos relacionados con el problema que se trabaja en esta tesis. En el capítulo **Métodos propuestos** se detalla la deducción y definición de los criterios de selección que se proponen. El capítulo de **Evaluación experimental** define el marco de cómo se pondrán a prueba los distintos criterios, además de presentar los resultados obtenidos. Finalmente, en **Conclusión** se discuten los resultados obtenidos en el capítulo anterior y se concluye el trabajo, además de plantearse distintas aristas para explorar posibles extensiones de este trabajo.

1.1. Hipótesis

Dada una base de datos de secuencias genómicas, el uso de sketches, como Hyperloglog [10] o SuperMinHash [8], permitirá establecer criterios de selección para encontrar el conjunto de pares de secuencias cuyo coeficiente de Jaccard supere un umbral $h \in (0, 1)$. Estos criterios reducirán el tiempo de cómputo y serán más rápidos y consistentes que el criterio directo (Ver definición 4.2.1). Dada la naturaleza de los sketches, se permite cierto margen de error y uso adicional de memoria para encontrar estos conjuntos, siempre que estos sean controlados.

1.2. Objetivos

1.2.1. Objetivo general

Proponer y analizar criterios de selección para reducir el número de comparaciones de similitud genómica cuando se necesita seleccionar pares de secuencias cuyo coeficiente de Jaccard sea mayor a un umbral $h \in (0, 1)$. Siempre con la finalidad de reducir el tiempo de cómputo.

1.2.2. Objetivos específicos

O1: Proponer criterios de selección basados en los sketches Hyperloglog y SuperMinHash a través de un análisis sobre la estimación de cardinalidad y de similitud, respectivamente.

O2: Implementar y evaluar los criterios propuestos con conjuntos pequeños de datos de la base de datos RefSeq[1]. Comparando los resultados con el criterio directo y con la ausencia de criterio (procesar todas las similitudes entre pares de secuencias posibles).

O3: Comparar, en distintos tipos de conjuntos, los resultados dados por los criterios propuestos con el criterio directo y con la ausencia de criterio (procesar todas las similitudes entre pares de secuencias posibles), además de compararlos entre sí.

1.3. Metodología de trabajo

Para la realización de este trabajo se contemplan las siguientes etapas:

I. **Revisión bibliográfica**: Entender el contexto del problema y el trabajo previo realizado. En particular se estudiarán los siguientes puntos:

   - Similitud genómica: Revisar cómo se estudia la similitud genómica, qué algoritmos existen y cuáles son las dificultades que enfrentan. Investigar

[1] https://www.ncbi.nlm.nih.gov/refseq/

sobre las métricas o coeficientes relacionados a la similitud genómica. Entender el pipeline desde que se obtiene una secuencia hasta que se mide su similitud con otra secuencia.

- Sketches y algoritmos probabilistas: Muchas de las estimaciones dependen de estructuras probabilistas llamadas sketches. Se revisarán los sketches que utilizan los algoritmos de similitud genómica además de buscar sketches adicionales que puedan aportar a este estudio.
- Criterios de selección: El objetivo de este trabajo es definir criterios de selección, por la misma razón se investigará sobre criterios de selección que se hayan aplicado para revisar cómo funcionan y tener una base con qué comparar los resultados. También se estudiará sobre métodos que permitan reducir el costo de encontrar o aproximar el conjunto $\mathcal{S}(\mathcal{M})$ definido en el capítulo **Métodos propuestos**.

II. **Análisis teórico**: En esta etapa se desarrollará el análisis teórico para la construcción de los criterios de selección. En particular, se desarrollará la deducción del criterio, se definirá formalmente y además, se realizará el análisis de error teórico según corresponda. Se utilizarán los sketches de SuperMinHash y de Hyperloglog para construir diferentes criterios.

III. **Recolección de datos**: Se extraerán secuencias genómicas de la base de datos de RefSeq para realizar los experimentos. Se buscará tener conjuntos variados de datos, con la finalidad de evaluar la robustez de los criterios frente a distintos tipos de datos. En particular, se tendrán genomas organizados en las siguientes clasificaciones: bacterias, archaea, fungi, plantas, protozoos, mamíferos vertebrados, otros vertebrados y viral. También se considerarán conjuntos de datos mezclando estas categorías.

IV. **Implementación**: Se realizará la implementación de los criterios propuestos en el **Análisis teórico**. Para ello se utilizará un servidor con sistema operativo Linux del departamento de Ingeniería Eléctrica de la Universidad de Concepción. El código será escrito en C++14 y se utilizarán bibliotecas disponibles

para implementar los distintos sketches. En caso de no existir la biblioteca, se implementarán directamente.

V. **Evaluación**: La evaluación se dividirá en dos etapas.

   a) Primero se evaluarán individualmente los criterios utilizando métricas de evaluación de problemas de clasificación como la sensibilidad y la especificidad. Se buscará que el criterio presente una sensibilidad cercana a 1, pues no se quieren descartar pares similares.

   b) Luego, además de comparar las métricas antes mencionadas entre los distintos criterios, también se realizará una comparación en cuanto al impacto que tienen los criterios en el uso de memoria y en el tiempo de ejecución, para así discutir sus aplicaciones prácticas.

   La evaluación se realizará sobre los conjuntos mencionados en **Recolección de datos**.

VI. **Presentación de resultados**: Terminada la evaluación, será necesario discutir y presentar los resultados obtenidos. Para ello se desarrollarán tablas y gráficos que permitan entender los resultados obtenidos en la evaluación, además de un análisis respecto del porqué se obtuvieron estos resultados.

VII. **Desarrollo de informe y escritura de artículo**: La investigación será presentada en un informe. Este incluirá lo obtenido de las etapas antes explicadas y describirá formalmente todo el proceso de investigación. Además, se redactará un articulo para presentar los resultados obtenidos con la finalidad de enviarlo a una revista afín.

Capítulo 2

Marco teórico

2.1. Conceptos generales

Un multiconjunto es una generalización de la idea de conjunto que permite la existencia de elementos repetidos. A diferencia de un conjunto convencional, donde cada elemento puede aparecer solo una vez, en un multiconjunto un elemento puede repetirse varias veces.

Formalmente, un multiconjunto se puede definir como un par $M = (A, m)$, donde A es el conjunto de elementos distintos de M, y $m : A \to \mathbb{N}$ es una función que mapea cada elemento $a \in A$ a su multiplicidad (número de ocurrencias de a en M) $m(a)$. Al conjunto A le llamamos conjunto subyacente de M. La cardinalidad de M (la cantidad de elementos distintos de M) se denotará por $|M|$. Notar que $|M|$ coincide con la cantidad de elementos distintos de A, es decir $|M| = |A|$.

Un multiconjunto $M = (A, m)$ se puede representar a través de su multiplicidad como el siguiente conjunto:

$$M = \{(a, m(a)) \ : \quad a \in A)\}.$$

Por ejemplo, $M = \{1, 1, 2, 5, 3, 2, 1, 5, 4\}$ es un multiconjunto, el cual también se representa como $M = \{(1, 3), (2, 2), (3, 1), (4, 1), (5, 2)\}$, ya que el 1 aparece 3 veces,

el 2, 2 veces, y así hasta el 5 que aparece 2 veces. Además, su cardinalidad es $|M| = 5$.

El alfabeto de los nucleótidos (unidades fundamentales del material genético) se denota por Σ, es decir, $\Sigma = \{A, C, G, T\}$. Σ^k denota al conjunto de todas las palabras de largo k sobre Σ, y Σ^*, al conjunto de todas las palabras de largo finito sobre Σ (sin importar su largo). Es decir, Σ^k corresponde a todas las cadenas de k nucleótidos y Σ^*, al conjunto de secuencias genómicas. Además, diremos que $w \in \Sigma^*$ es sub-palabra de $L \in \Sigma^*$, si los símbolos de w aparecen en L de forma contigua y en el mismo orden. Notar que el largo de w no puede ser superior al de L. Por ejemplo, $L = ACTGCCGT \in \Sigma^*$ es una secuencia genómica de largo 8 y $w = CTG \in \Sigma^3$ es una sub-palabra de L ya que aparece entre el segundo y cuarto símbolo.

Dados una secuencia genómica $L \in \Sigma^*$, y un número $k \in \mathbb{N}$, un k-mer de L corresponde a una sub-palabra $a \in \Sigma^k$ de L. El conjunto de k-mers de L se denota por $D_k(L)$, es decir,

$$D_k(L) = \{w \in \Sigma^k \ : \ w \text{ es sub-palabra de } L\}.$$

Por otro lado, el multiconjunto de k-mers de L, que denotaremos por $spec_k(L)$, corresponde al multiconjunto $(D_k(L), m)$, siendo $m : D_k(L) \to \mathbb{N}$ la función que retorna la multiplicidad de cada k-mer en L. Es decir,

$$spec_k(L) := \{(a, m(a)) \ : \quad a \in D_k(L)\}.$$

La cardinalidad de este multiconjunto se denotará por $d_k(L)$. Por ejemplo, si $L = ATTTCGTC$, entonces

$$\begin{aligned} spec_2(L) &= \{AT, TT, TT, TC, CG, GT, TC\} \\ &= \{(AT, 1), (TT, 2), (TC, 2), (CG, 1), (GT, 1)\}, \end{aligned}$$

y $d_2(L) = 5$, ya que sólo aparecen 5 k-mers distintos.

Notar que si L es una secuencia genómica de largo $N \in \mathbb{N}$, entonces, para todo $k \in \mathbb{N}$ se cumple que:

$$d_k(L) \leq \text{mín}\{4^k, N - k + 1\}.$$

Es evidente que $d_k(L) \leq 4^k$, pues $D_k(L) \subset \Sigma^k$ y $|\Sigma^k| = 4^k$. Además, como la secuencia L es de largo N, entonces contiene $N-k+1$ sub-palabras de largo k (no necesariamente distintas), así que $d_K(L) \leq N-k+1$.

Dados dos multiconjuntos A, B, el coeficiente de Jaccard entre A y B, denotado por $J(A,B)$, corresponde a la porción de elementos en común entre A y B, sin contar sus multiplicidades. Esto es,

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A| + |B| - |A \cup B|}{|A \cup B|},$$

donde la segunda igualdad está dada por el principio de inclusión-exclusión $|A \cup B| = |A| + |B| - |A \cap B|$.

Para aplicaciones basadas en distancias en lugar de similitud, el coeficiente de Jaccard puede ser transformado a la distancia de Jaccard $J_\delta = 1 - J$ [18].

En el caso particular de secuencias, dadas dos secuencias $L_1, L_2 \in \Sigma^*$, $J_k(L_1, L_2)$ denota el coeficiente de Jaccard entre $spec_k(L_1)$ y $spec_k(L_2)$. Es decir,

$$\begin{aligned} J_k(L_1, L_2) &= J(spec_k(L_1), spec_k(L_2)) \\ &= \frac{|spec_k(L_1)| + |spec_k(L_2)| - |spec_k(L_1) \cup spec_k(L_2)|}{|spec_k(L_1) \cup spec_k(L_2)|} \\ &= \frac{|D_k(L_1)| + |D_k(L_2)| - |D_k(L_1) \cup D_k(L_2)|}{|D_k(L_1) \cup D_k(L_2)|} \end{aligned}$$

2.2. Teorema de Taylor

El teorema de Taylor es un teorema que permite aproximar una función en un punto donde sea diferenciable por medio de un polinomio.

Teorema 2.2.1 (Teorema de Taylor)**.** *Sea $n \in \mathbb{N}$ y sea $f : \mathbb{R} \to \mathbb{R}$ una función diferenciable n veces en el punto $a \in \mathbb{R}$. Entonces existe una función $h_n : \mathbb{R} \to \mathbb{R}$ tal que*

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + ... + \frac{f^{(n)}(a)}{n!}(x-a)^n + h_n(x)(x-a)^n,$$

con $\lim_{x \to a} h_n(x) = 0$.

El polinomio que aparece en el teorema se denomina polinomio de Taylor de orden n de la función f en el punto a, y se denota por $P_n(x)$. Es decir,

$$P_n(x) = \sum_{i=0}^{n} \frac{f^{(i)}(a)}{i!}(x-a)^i,$$

siendo $f^{(0)} = f$.

El término del resto, denotado por $R_n(x)$, corresponde al error de aproximación de $f(x)$ dado por $P_n(x)$, es decir,

$$R_n(x) = f(x) - P_n(x).$$

Suponiendo que f es continuamente diferenciable $n+1$ veces en un intervalo I que contiene a a, y suponiendo que existen constantes q y Q tales que

$$q \leq f^{(n+1)}(x) \leq Q$$

en el intervalo I. Entonces se cumple que:

$$q\frac{(x-a)^{n+1}}{(n+1)!} \leq R_n(x) \leq Q\frac{(x-a)^{n+1}}{(n+1)!}.$$

En particular, si

$$|f^{(n+1)}(x)| \leq M,$$

sobre un intervalo $I = (a-r, a+r)$, con $r > 0$, entonces

$$|R_n(x)| \leq M\frac{|x-a|^{n+1}}{(n+1)!} \leq M\frac{r^{n+1}}{(n+1)!}. \tag{2.2.1}$$

2.3. Funciones de hash

El hashing [5] es una técnica clásica para insertar, borrar y buscar con tiempo promedio constante. La idea es construir una función de hash $h : \mathcal{D} \to \{0, 1, ..., m-1\}$ que mapee los objetos de su universo original $\mathcal{D}$ a un valor en $\{0, 1, ..., m-1\}$. Así, los elementos $x \in \mathcal{D}$ se corresponden inequívocamente a un índice en $\{0, 1, ..., m-1\}$, a esta correspondencia se le conoce como tabla de hash. De esta forma, buscar un

elemento en esta tabla es tan costoso como computar su función de hash, lo cual en la práctica es $\mathcal{O}(1)$.

Un problema que pueden presentar las funciones de hash son las colisiones, esto es, cuando dos valores distintos $x, y \in \mathcal{D}$ se mapean al mismo índice ($h(x) = h(y)$). Esto provoca que existan elecciones de elementos en $\mathcal{D}$ de forma que ocurran muchas colisiones, perdiéndose las garantías de tiempo $\mathcal{O}(1)$. Para lidiar con esto, en la práctica se utiliza el hashing universal.

Definición 2.3.1 (Hashing Universal [5])**.** *Sea $\mathcal{D}$ nuestro universo finito de elementos. Sea también $\mathcal{H}$ una familia de funciones hash que mapean los elementos de $\mathcal{D}$ a $\{0, 1, ..., m-1\}$, con $m \in \mathbb{N}$. Se dice que $\mathcal{H}$ es familia universal de funciones de hash si para todo par $x, y \in \mathcal{D}, x \neq y$, se tiene que*

$$Pr(h(x) = h(y)) \leq \frac{1}{m},$$

donde la probabilidad se toma con respecto a las posibles elecciones de funciones $h \in \mathcal{H}$ (eligiendo al azar y uniformemente).

De esta forma, al tomar aleatoriamente una función de hash $h \in \mathcal{H}$, la probabilidad de colisión entre dos elementos distintos $x, y \in \mathcal{D}$ no es más que la probabilidad de colisión si $h(x)$ y $h(y)$ se hubieran escogido aleatoria e independientemente sobre los valores $\{0, 1, ..., m-1\}$, lográndose así un control sobre las colisiones.

2.4. Sketches y streaming

Sea X un conjunto de datos y $f(X)$ alguna función de interés que se quiera estimar, como por ejemplo la moda de X. Un sketch $S(X)$ de X es una estructura de datos, generalmente probabilística, que resume a X de forma que se pueda estimar $f(X)$ a partir de $S(X)$. Esta estructura de datos utiliza espacio acotado, típicamente sublineal.

Dado que el objetivo es estimar $f(X)$, estas estructuras de datos suelen justificarse a través de la teoría de la probabilidad. Uno de los primeros algoritmos de esta clase [9] utiliza variables geométricas para estimar la cantidad de elementos distintos en

un conjunto X. Esta idea es justamente la que entrega la intuición para el sketch principal que se revisará en esta memoria: el sketch de Hyperloglog.

Teniendo X, siempre es posible obtener $f(X)$ de forma exacta si no se tienen restricciones del espacio ocupado. Sin embargo, los conjuntos de datos hoy en día pueden llegar a ser gigantescos, además de estar actualizándose constantemente. Por ejemplo, el tráfico de un sitio web como Google puede rondar los 80 billones mensuales en la actualidad. Otro ejemplo, relacionado directamente con este trabajo, es el de la base de datos RefSeq, la cual aumenta en más de 30000 secuencias genómicas al año [13]. Realizar mediciones sobre conjuntos tan grandes puede requerir mucho espacio. En el caso de calcular cardinalidad (cantidad de individuos distintos que visitan Google), se necesita almacenar cada elemento distinto. Tanto la cantidad de memoria necesaria, como el acceso a esta, provocan que este cálculo sea muy costoso computacionalmente.

El concepto de streaming viene de la idea de construir $S(X)$ realizando una inspección secuencial sobre los elementos de X, además, X suele ser un conjunto que se actualiza constantemente. Así, es deseable poder actualizar $S(X)$ a medida que nuevos elementos son agregados a X. En la literatura los sketches son frecuentemente asociados a una clase de algoritmos de streaming.

2.4.1. Sketch Hyperloglog

Sea X un multiconjunto de cardinalidad $n \in \mathbb{N}$ con elementos de un universo $\mathcal{U}$. El sketch de Hyperloglog [10], [14] consiste en un arreglo S de tamaño $m = 2^p$, con $p \in \mathbb{N}$, que se construye según el Algoritmo 1.

El arreglo S resultante de procesar X permite estimar la cardinalidad de X. Esta estimación E está dada por

$$E = \alpha_m m^2 \left(\sum_{j=1}^{m} 2^{-S[j]} \right)^{-1},$$

Algoritmo 1 Algoritmo de construcción del sketch Hyperloglog

Entrada: Multiconjunto X de cardinalidad desconocida y $p \in \mathbb{N}$.

1: Inicializar un arreglo S con $m = 2^p$ registros $S[1], S[2], ..., S[m]$ en $-\infty$.
2: **for** $x \in X$ **do**
3: $\quad b \leftarrow h(x)$
4: $\quad j \leftarrow 1 + \langle b_1 b_2 ... b_p \rangle_2$ ▷ Índice determinado por los primeros p bits de b
5: $\quad w \leftarrow b_{p+1} b_{p+2} ...$
6: $\quad S[j] \leftarrow \max(S[j], \rho(w))$
7: **end for**

return: S

donde α_m está dado por (2.4.1).

$$\alpha_m := \left(m \int_0^\infty \left(\log \frac{2+u}{1+u} \right)^m \right)^{-1}. \tag{2.4.1}$$

La figura 2.1 muestra cómo funciona el algoritmo de Hyperloglog detallado anteriormente. Primero se toman los elementos e pertenecientes a un multiconjunto o *stream* (flujo de datos) y se calcula su valor de hash $h(e)$. Este valor se separa en los primeros p bits que representan un valor v_1 y en los últimos b bits que representan un valor v_2. Luego, se actualiza la posición v_1 del sketch (o arreglo) A por el máximo entre el valor actual de $A[v_1]$ y $\rho(v_2)$, donde ρ es una función que retorna la posición del primer bit igual a 1, de izquierda a derecha, de la secuencia evaluada. Una vez procesados los elementos del multiconjunto, se estima su cardinalidad utilizando los valores de A.

El respaldo teórico de la estimación dada por este sketch está dado por el siguiente teorema:

Teorema 2.4.1. *Sea X un multiconjunto de cardinalidad n desconocida. Y sea E el resultado de aplicar el algoritmo de Hyperloglog a X utilizando $m \in \mathbb{N}$ registros según lo descrito anteriormente. Entonces*

1. *E es asintóticamente casi insesgado, esto es:*

$$\frac{1}{n}\mathbb{E}_n[E] \xrightarrow{n\to\infty} 1 + \delta_1(n) + o(1).$$

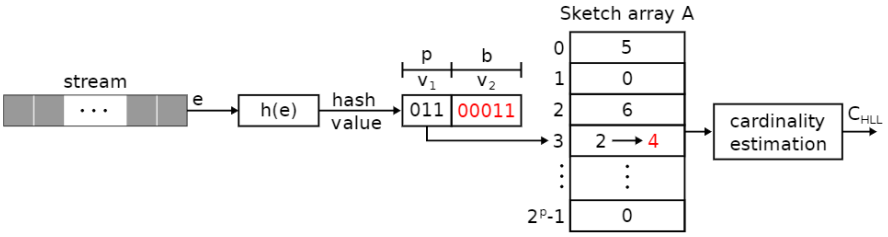


Figura 2.1: Esquema general de Hyperloglog. Fuente: [25]

2. *El error estándar de E, dado por $\frac{1}{n}\sqrt{\mathbb{V}_n[E]}$, cumple lo siguiente:*

$$\frac{1}{n}\sqrt{\mathbb{V}_n[E]} \xrightarrow{n\to\infty} \frac{\beta_m}{\sqrt{m}} + \delta_2(n) + o(1).$$

Donde

- *δ_1 y δ_2 son funciones oscilantes de amplitud muy pequeña, aproximadas a 0 para efectos prácticos.*
- *La constante β_m es acotada, con $\beta_{16} = 1.106$, $\beta_{32} = 1.070$, $\beta_{64} = 1.054$, $\beta_{128} = 1.06$, y $\beta_\infty = \sqrt{3\log 2 - 1} = 1.03896$.*

Así, se espera que el estimador se acerque a n y, al aumentar el tamaño del sketch, se espera que el error estándar disminuya. Se denotará por $|X|_p$ a la estimación de cardinalidad obtenida por un sketch de Hyperloglog de tamaño 2^p.

Sean X, Y dos multiconjuntos, $x \in X$ e $i \in \{0, 1, ..., 2^p - 1\}$, y sean S_X y S_Y los respectivo sketches de Hyperloglog de tamaño 2^p de X e Y. Dada la forma en que se procesan los elementos de X para construir S_X, es posible establecer las siguientes propiedades del sketch de Hyperloglog:

$$i)\ |X \cup \{x\}|_p = |X|_p, \qquad \text{(Idempotente)}$$

$$ii)\ S_{X\cup Y}[i] = \text{máx}\{S_X[i], S_Y[i]\}, \qquad \forall i \in \{1, 2, ..., m\}. \qquad \text{(Fusionable)}$$

Donde $S_{X\cup Y}$ denota al sketch de Hyperloglog de tamaño 2^p asociado a $X \cup Y$. Es decir, insertar más de una vez un elemento en el sketch tiene el mismo efecto en la estimación de cardinalidad que haberlo insertado una única vez, claramente esta es una propiedad deseable para la estimación de cardinalidad. Por otro lado, la propiedad de fusión o merge, establece que a través de los sketches asociados a X e Y es posible construir el sketch asociado a $X \cup Y$, sin necesidad de procesar la unión como tal.

El error relativo estándar de un Hyperloglog de tamaño 2^p (es decir, $m = 2^p$) es $\sigma_m = \beta_m/\sqrt{m}$, con β_m definido en el teorema 2.4.1. Según [10] se espera que las estimaciones de cardinalidad entregadas por Hyperloglog estén a σ_m, $2\sigma_m$ y $3\sigma_m$ del valor exacto de cardinalidad en el 65 %, 95 %, 99 % de todos los casos, respectivamente. Es decir,

$$|X|_p \in [|X| - Z_\alpha \sigma_m |X|, |X| + Z_\alpha \sigma_m |X|]\,, \tag{2.4.2}$$

en el α % de los casos, donde $\alpha \in \{65, 95, 99\}$ y $Z_\alpha \in \{1, 2, 3\}$ es el respectivo factor que acompaña a σ_m.

Por comodidad, se usará la notación σ_p para referirse a σ_m. Es decir,

$$\sigma_p := \frac{\beta_{2^p}}{\sqrt{2^p}}.$$

2.4.2. Sketch SuperMinHash

El algoritmo de MinHash [22, 4] busca estimar el índice de Jaccard entre dos conjuntos. Su fundamento se encuentra en el siguiente teorema

Teorema 2.4.2. *Sea $\mathcal{U} = \{1, 2, ..., n\}$ y sea $\mathcal{P}_n$ el conjunto de permutaciones sobre $\{1, 2, ..., n\}$. Entonces, si $\pi \in \mathcal{P}_n$ es escogida uniformemente de forma aleatoria, se tiene que*

$$\forall A, B \subset \mathcal{U}\ ,\ \ Pr[\text{mín}(\pi(A)) = \text{mín}(\pi(B))] = J(A, B).$$

Notar que este resultado se extiende para un universo $\mathcal{U}$ finito de tamaño n cualquiera, para ello basta considerar una indexación $idx : \mathcal{U} \to \{1, 2, ..., n\}$ y aplicar el

mínimo sobre $(\pi \circ idx)(A)$ y $(\pi \circ idx)(B)$.

De esta forma, dados dos conjuntos A, B, si realizamos varias permutaciones arbitrarias, es posible estimar la probabilidad $Pr[\text{mín}(\pi(A)) = \text{mín}(\pi(B))]$ y, por lo tanto, estimar la similitud $J(A, B)$ según el teorema 2.4.2. En la práctica, realizar permutaciones resulta costoso computacionalmente, es por ello que se recurre al uso de funciones de hash para simular una permutación cualquiera. De esta forma, suponiendo que se utilizan k funciones de hash $h_1, h_2, ..., h_k : \mathcal{U} \to \{1, 2, ..., n\}$ (para simular k permutaciones), por cada conjunto $A \subset \mathcal{U}$ se construye su firma $S(A)$ (sketch) como el arreglo de tamaño k dado por

$$\forall i \in \{1, 2, ..., k\} \ , \quad S(A)[i] = \text{mín}(h_i(A)).$$

Así, dados dos conjuntos A, B, basta comparar $S(A)$ con $S(B)$ para estimar $J(A, B)$.

En general, la estrategia es aproximar el efecto de $\pi \circ idx : \mathcal{U} \to \{1, 2, ..., n\}$ a través de funciones de hash $h : \mathcal{U} \to \{1, 2, ..., n\}$. En la práctica, es posible construir funciones de hash que imiten suficientemente bien el comportamiento de $\pi \circ idx$.

SuperMinHash [8] es el estado del arte con respecto a estimación del coeficiente de Jaccard. Este algoritmo mantiene la naturaleza general de MinHash, es decir, se construye un arreglo a través del cual se estima el coeficiente de Jaccard entre multiconjuntos, pero se demuestra que dada la nueva construcción del sketch, la varianza del estimador es estrictamente menor que en el caso de MinHash clásico. Este mejora se debe a una componente adicional de permutación (representada por la variable p en el algoritmo 2), lo cual lleva a que los elementos resultantes en el arreglo no sean independientes.

El algoritmo 2 muestra la construcción del sketch de SuperMinHash. En el artículo [8], además de este algoritmo, presentan otra versión con optimizaciones adicionales en cuanto a la implementación.

Algoritmo 2 Algoritmo de construcción del sketch SuperMinHash

Entrada: Multiconjunto X y $m \in \mathbb{N}$.

Inicializar un arreglo S con m registros $S[1], S[2], ..., S[m]$ en ∞.
Inicializar un arreglo auxiliar p con m registros.
Inicializar un arreglo auxiliar q con m registros $q[1], q[2], ..., q[m]$ en -1.
$i \leftarrow 0$.
for $x \in X$ **do**
    Inicializar generador de números pseudo-aleatorio con semilla x.
    $p \leftarrow (0, 1, ..., m-1)$.
    for $j \leftarrow 0, 1, ..., m-1$ **do**
        $r \leftarrow$ número aleatorio uniforme de $[0, 1)$.
        $k \leftarrow$ número aleatorio uniforme de $\{j, j+1, ..., m-1\}$.
        if $q[j] \neq i$ **then**
            $q[j] \leftarrow i$
            $p[j] \leftarrow j$
        end if
        if $q[k] \neq i$ **then**
            $q[k] \leftarrow i$
            $p[k] \leftarrow k$
        end if
        $swap(p[j], p[k])$
        $S[p[j]] \leftarrow \text{mín}(S[p[j]], r + j)$
    end for
    $i \leftarrow i + 1$
end for

return: S

2.5. Local Sensitive Hashing

Local Sensitive Hashing (LSH) para MinHash es un enfoque que busca encontrar aquellos pares de elementos cuyo coeficiente de similitud sea mayor a un umbral, para ello se buscan pares que tienen suficiente probabilidad de ser similares, evitando comparar todos los posibles pares. Una de las formas generales de aplicar este enfoque es utilizando funciones de hash varias veces de modo tal que elementos similares tienen más probabilidades de terminar colisionando entre si (resultar en el mismo valor de hash). De esta forma los pares que colisionen serán seleccionados como pares candidatos y el cálculo de similitud se realizará sólo sobre los pares candidatos, con la esperanza de que la menor cantidad de pares disimilares sean seleccionados como pares candidatos, y que la mayoría de los pares similares sí lo sean.

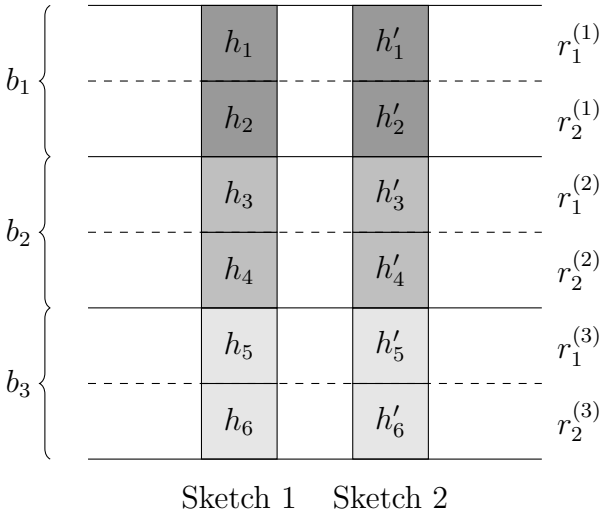


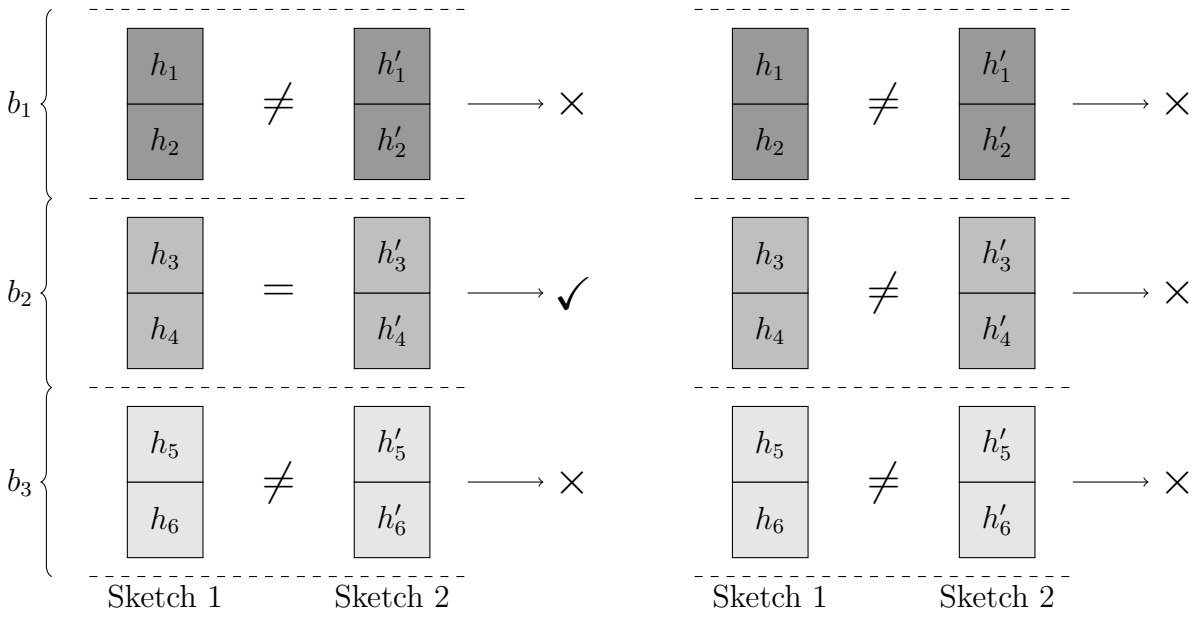
Figura 2.2: Partición de sketch de SuperMinHash de tamaño $m = 6$, utilizando $b = 3$ bandas y $r = 2$ filas.

El sketch SuperMinHash, dadas sus propiedades, permite definir una estrategia que cumple con este enfoque. Sean A, B dos multiconjuntos y consideremos los sketches de SuperMinHash $S_m(A)$ y $S_m(B)$ de A y B, respectivamente, ambos de tamaño $m \in \mathbb{N}$. Luego,

$$\forall i \in \{0, 1, 2, ..., m-1\}\ , \quad Pr(S_m(A)[i] = S_m(B)[i]) = J(A, B).$$

Sean $r, b \in \mathbb{N}$ tales que $rb = m$. Esto genera una partición de b bandas y r filas, es decir, el arreglo de SuperMinHash se particiona en b conjuntos de buckets, cada uno con r buckets contiguos. Por ejemplo, en la figura 2.2 se muestran dos sketches de SuperMinHash de tamaño $m = 6$, los cuales fueron divididos en $b = 3$ bandas b_1, b_2, b_3, cada una con $r = 2$ filas $r_1^{(i)}$ y $r_2^{(i)}$, con $i = 1, 2, 3$ el índice de la banda correspondiente.

Por simplicidad, denotemos $s = J(A, B)$. Luego, la probabilidad de que las r filas sean iguales en alguna de las b bandas entre los dos sketches de SuperMinHash es igual a s^r, entonces la probabilidad de que los sketches no coincidan en alguna de las bandas es $1 - s^r$. De esta forma, la probabilidad de que no coincidan en las b bandas es $(1 - s^r)^b$ y, finalmente, la probabilidad de que coincidan en alguna de las bandas es $1 - (1 - s^r)^b$. Notar que esta probabilidad es creciente con respecto a s, es decir, mientras más alto sea el valor de $J(A, B)$, más probable es que coincidan en alguna de las bandas. Así, la técnica de LSH utilizando el sketch de SuperMinHash consiste en seleccionar como pares candidatos a aquellos pares que coincidan en alguna de las bandas de sus respectivos arreglos de SuperMinHash. En la figura 2.3 se muestran un ejemplo de cómo funciona esta técnica considerando 3 bandas, cada una de 2 filas.



(a) Si los arreglos coinciden en alguna de las bandas, entonces el par de secuencias se marca como par candidato.

(b) Si los arreglos no coinciden en ninguna de las bandas, entonces el par de secuencias no es un par candiato.

Figura 2.3: Esquema de funcionamiento de LSH usando sketches de SuperMinHash de tamaño $m = 6$. El símbolo ✓ denota que la igualdad se cumple, y el símbolo $\times$, que no.

Capítulo 3

Estado del arte

3.1. Comparación de genomas

Cerca del año 1990 fueron desarrollados múltiples herramientas para trabajar con el alineamiento de secuencias, como BLAST y FASTA [28]. Estas herramientas facilitarían el trabajo de computar medidas de similitud basadas en alineamiento como lo es la Identidad Nucleótida Media. Si bien este era el estándar, una de las grandes dificultades de los métodos basados en alineamiento es el consumo de memoria y tiempo [28], en especial con el tamaño actual de las bases de datos. Para enfrentar esta dificultad surgen los métodos o análisis de secuencias genómicas libres de alineamiento.

Los métodos libres de alineamiento se definen como métodos que computan la similitud o distancia entre secuencias sin utilizar o producir un alineamiento entre las secuencias en ninguno de sus pasos. Estos métodos se pueden categorizar en dos amplias categorías [28]: métodos basados en frecuencias de sub-palabras y métodos basados en la teoría de la información.

Este trabajo se enmarca en el contexto de métodos libres de alineamiento y basados en la frecuencia de sub-palabras, ya que se utilizan los k-mers de una secuencia para estudiar, a través del coeficiente de Jaccard, la similitud entre estas. Hay dos

herramientas que utilizan esta metodología para trabajar con secuencias: MASH y Dashing, con la diferencia en el sketch que utilizan para estimar el coeficiente de Jaccard.

3.1.1. MASH

MinHash es un sketch que recibe un multiconjunto de datos X y, por medio de funciones de hash, construye un arreglo (sketch) $S(X)$ para identificar a X. Luego, si se tienen los sketches de distintos multiconjuntos X_1, X_2, entonces comparar los sketches $S(X_1)$ y $S(X_2)$ permite computar una aproximación j de $J(X_1, X_2)$ (Ver Figura 3.1).

Mash [21] es una herramienta que utiliza como base el sketch de MinHash para procesar secuencias genómicas y calcular la distancia Mash. La llamada distancia Mash, denotada por D, entre dos secuencias $S_1, S_2 \in \Sigma^*$ está dada por

$$D(S_1, S_2) = -\frac{1}{k} \ln \frac{2j}{j+1}, \tag{3.1.1}$$

donde k es el parámetro para generar los k-mers y j es la estimación del coeficiente de Jaccard entre $spec_k(S_1)$ y $spec_k(S_2)$ entregada por los sketches de MinHash. D no es una distancia o métrica en el sentido matemático, pues no cumple la desigualdad triangular.

Para la deducción de esta distancia se considera la probabilidad d de que ocurra una mutación puntual (que una letra de la secuencia cambie) en un genoma w. Luego, bajo un modelo de Poisson, la probabilidad de que no ocurra ninguna mutación en un k-mer dado es de e^{-kd}, cuyo valor esperado corresponde a la fracción de k-mers no mutados w con respecto al número total de k-mers t en el genoma. Resolviendo $e^{-kd} = \frac{w}{t}$ resulta en $d = -\frac{1}{k} \ln \frac{w}{t}$. Para tener en cuenta el tamaño de ambos genomas a comparar, se considera t como el promedio de los tamaños de ambas secuencias, denotado por n. Finalmente, como la estimación del Jaccard j puede ser expresada en términos del tamaño promedio de los genomas $j = \frac{w}{2n-w}$, la fracción de k-mers compartidos puede ser expresada en términos de j, $\frac{w}{n} = \frac{2j}{j+1}$. Obteniéndose la distancia Mash (3.1.1).

Así, la distancia Mash D busca capturar la tasa de mutación para que de una secuencia se obtenga la otra. Por lo que un bajo valor de D indica que la tasa de mutación debe ser baja, es decir, las secuencias son muy similares. Por el contrario, si D es grande, entonces se infiere una alta tasa de mutación, indicando que las secuencias no son muy similares.

La relación entre el coeficiente de Jaccard J y la distancia Mash M es clara (D se define a partir de una estimación de J). Además, se muestra también en [21] que D se correlaciona también con el ANI. En específico, D se aproxima a $1-\text{ANI}$. Así, queda justificada la relevancia de estimar J o D para ahorrar recursos en la obtención del ANI.

3.1.2. Dashing

La estimación de MinHash presenta problemas al comparar conjuntos de tamaños muy diferentes [17]. Este escenario no es tan extraño, por ejemplo al comparar el genoma de una bacteria, que es muy pequeño, a una colección de meta-genomas (muchos genomas relacionados) [1].

Dashing [1] es una herramienta que utiliza el sketch de Hyperloglog para estimar la cantidad de k-mers distintos en una secuencia, y así estimar el coeficiente de Jaccard a partir de las cardinalidades correspondientes. Hyperloglog exhibe una excelente precisión y rapidez en una amplia variedad de escenarios, incluso cuando los conjuntos de entrada son de tamaños muy diferentes o cuando los sketches correspondientes son pequeños [3].

Dadas dos secuencias L_1, L_2 y un valor $k \in \mathbb{N}$, y siguiendo la notación establecida en esta tesis, el coeficiente de Jaccard entre sus respectivos conjuntos de k-mers $D_k(L_1)$ y $D_k(L_2)$, se puede calcular como

$$J_L(L_1, L_2, k) = \frac{|D_k(L_1)| + |D_k(L_2)| - |D_k(L_1) \cup D_k(L_2)|}{|D_k(L_1) \cup D_k(L_2)|}. \tag{3.1.2}$$

Así, se puede construir los sketches de Hyperloglog de $S(spec_k(L_1))$ y $S(spec_k(L_2))$ asociados a L_1 y L_2, respectivamente. Luego, a partir de estos sketches se pueden es-

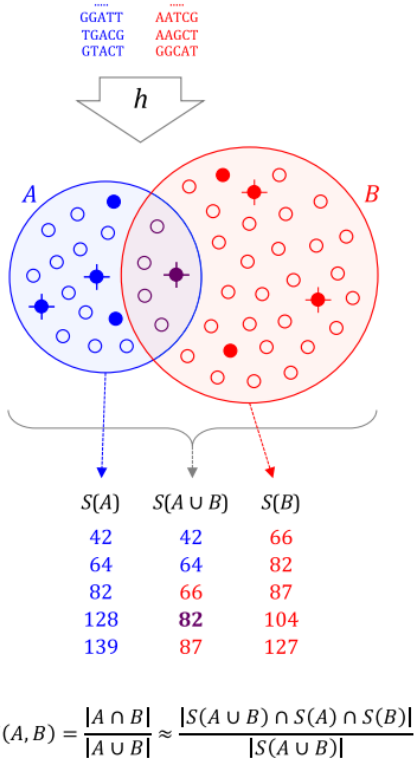


Figura 3.1: Resumen de la estrategia de MinHash para estimar J. Primero, cada k-mer de los respectivos multiconjuntos de k-mers de dos secuencias es procesado por una función de hash h de 32 o 64 bits. Los conjuntos de hashes resultantes A y B contienen $|A|$ y $|B|$ hashes distintos (círculos pequeños). El coeficiente de Jaccard corresponde entonces a la fracción de hashes en común (círculos púrpuras). Este valor puede aproximarse utilizando una muestra aleatoria de $A \cup B$ mucho más pequeña. Los sketches de MinHash $S(A)$ y $S(B)$ de tamaño 5 asociados a A y B, respectivamente, se muestran en la figura. Estos son formados por los 5 menores valores de hash en cada conjunto (círculos rellenos). Utilizar $S(A)$ y $S(B)$ para obtener los 5 menores valores de $A \cup B$ (círculos con cruz), resulta en $S(A \cup B)$. Como $S(A \cup B)$ corresponde a una muestra aleatoria de $A \cup B$, la fracción de elementos en $S(A \cup B)$ que son compartidos tanto por $S(A)$ como por $S(B)$ es un estimador insesgado de $J(A, B)$. Fuente: [21]

timar $|D_k(L_1)|, |D_k(L_2)|$ y $|D_k(L_1) \cup D_k(L_2)|$. Reemplazando estos valores en (3.1.2) se obtiene una estimación de $J(D_k(L_1), D_k(L_2))$. Dashing ocupa esta estrategia para computar estimaciones del coeficiente de Jaccard, además de ciertas correcciones al estimador de cardinalidad para mejorar el error en rangos de cardinalidad muy pequeños o muy grandes.

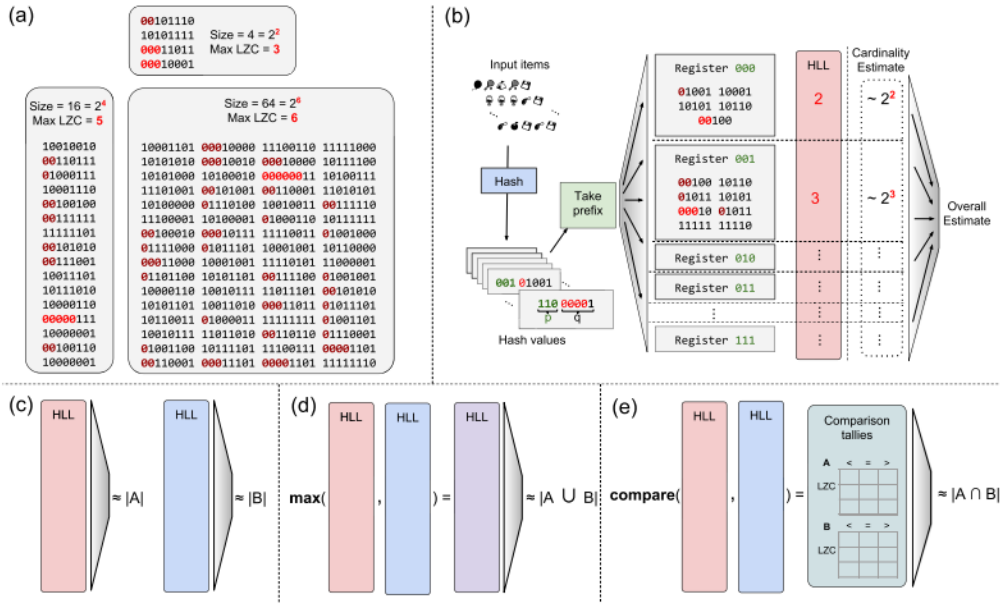


Figura 3.2: **(a)** Relación entre la máxima cantidad consecutiva de ceros a la izquierda (LMZ, Max Leading Zero Count) y la cardinalidad de conjuntos de números de 8 bits aleatoriamente generados. **(b)** Esquema de funcionamiento del sketch de Hyperloglog. **(c)** Estimación de la cardinalidad para los multiconjuntos A y B, y **(d)** estimación de la cardinalidad de su unión. Haciendo uso del principio de inclusión-exclusión es posible estimar la cardinalidad de la intersección. **(e)** Estimación directa de la cardinalidad de la unión utilizando el estimador JMLE de Ertl. Fuente: [1]

3.2. Cómo elegir k

Uno de los problemas claves del proceso de comparar dos secuencias genómicas es el de cómo elegir el valor de $k \in \mathbb{N}$ para generar los conjuntos de k-mers que se procesarán. Claramente una elección arbitraria puede llevar a resultados no significativos. Por ejemplo, si k es muy pequeño (por ejemplo $k = 1$), entonces $d_k(L)$ será muy cercano a 4^k para todas las secuencias L suficientemente largas. Por otro lado, si k es muy grande (cercano al largo de la misma secuencia por ejemplo), entonces $d_k(L)$ será muy cercano a 1.

3.2.1. Coeficiente de compresibilidad δ

Existen medidas que buscan comprimir secuencias capturando la repetividad que posean, una de ellas es la medida δ [16].

Definición 3.2.1. *Sea una secuencia L de largo n, se define*

$$\delta(L) = \text{máx}\{d_k(L)/k \ : \ k \in \{1, 2, ..., n\}\}.$$

Se obtiene así una medida que es independiente a la elección de k. Al recorrer los distintos valores de k se pueden observar tres etapas en el comportamiento de $d_k(L)/k$. Para valores de k que son muy pequeños, de modo que casi todas las k-mers aparecen en L, $d_k(L)/k$ crece exponencialmente. Para valores de k cercanos a n, $d_k(L)/k$ decrece linealmente, eventualmente llegando a 1 cuando $k = n$. Para valores intermedios de k, ambos comportamientos anteriores están en tensión, incrementar k aumenta la cantidad de posibles k-mers, pero eventualmente en la secuencia dejan de aparecer muchos nuevos k-mers. El k que maximiza $d_k(L)/k$ identifica este punto en el que se deja de obtener nueva información [2].

Si bien es costoso calcular $d_k(L)$ para varios valores de k, se pueden usar sketches para estimar su valor. Los autores de DanD [2] utilizan la herramienta de Dashing para construir los sketches y así calcular δ para distintas secuencias. DanD también ocupa esta medida de compresibilidad para proponer un nuevo coeficiente de similitud: el coeficiente de Jaccard independiente de k, KIJ por sus siglas en inglés.

Definición 3.2.2. *Sean L_1, L_2 dos secuencias. El coeficiente de Jaccard independiente de k entre L_1 y L_2, denotado por $KIJ(L_1, L_2)$, está dado por*

$$KIJ(L_1, L_2) = \frac{\delta(L_1) + \delta(L_2) - \delta(L_1 \cup L_2)}{\delta(L_1 \cup L_2)}.$$

Independizando el estudio de la similitud de secuencias del parámetro k. Además, realizando comparaciones con el coeficiente de Jaccard estándar, se muestra que KIJ es una buena medida para comparar similitud de secuencias. Sin embargo, los autores discuten que es necesario un estudio más profundo para ver cómo se compara KIJ con la distancia Mash que utiliza un valor de k en específico, pues al momento

de calcular $KIJ(L_1, L_2)$ no necesariamente se utilizan los mismos valores de k para calcular $\delta(L_1), \delta(L_2)$ y $\delta(L_1 \cup L_2)$.

3.3. Reducir número de comparaciones

Estimar similitud utilizando herramientas como Dashing o Mash, conlleva comparar sketches entre sí. Menos comparaciones resultan en menos tiempo de ejecución, pero la precisión de los resultados se puede ver afectada. A continuación se presentan resultados con respecto a la cantidad de comparaciones. En particular, se discute el tamaño de los sketches de Hyperloglog para estimar un nuevo coeficiente de inclusión y también el cómo utilizar los sketches que se construyen con Dashing para omitir directamente ciertas comparaciones entre sketches.

3.3.1. Estimación del coeficiente de inclusión Φ

Dados dos conjuntos A, B, el coeficiente de inclusión entre A y B, denotado por $\Phi(A, B)$ se define como

$$\Phi(A, B) = \frac{|A \cap B|}{|A|}, \qquad A \neq \emptyset.$$

Es decir, $\Phi(A, B)$ corresponde a la fracción de elementos en A que también están en B. Este coeficiente es bastante similar al coeficiente de Jaccard, de hecho, el coeficiente de Jaccard entre dos conjuntos X, Y corresponde al coeficiente de inclusión entre $X \cup Y$ e $X \cap Y$ (fracción de elementos en la unión que están en ambos conjuntos).

En este trabajo se busca estimar el coeficiente de inclusión Φ utilizando el sketch de Hyperloglog, además de establecer una relación entre el error de estimación de Φ y el tamaño del sketch utilizado.

Sean A, B dos conjuntos y $S_p(A)$, $S_p(B)$ sus respectivos sketches de Hyperloglog, ambos de tamaño $m = 2^p$. En [20] se demuestra que $P = Pr[S_1(A) \leq S_1(B)]$ es función creciente de $\Phi(A, B)$, donde $S_1(A)$ y $S_1(B)$ son sketches de Hyperloglog

de tamaño 1 de A y B, respectivamente, es decir, $S_1(A)$ es el valor que hay en la única casilla del sketch de Hyperloglog correspondiente, lo mismo para $S_1(B)$. Así, se pueden usar los sketches $S_p(A)$ y $S_p(B)$ para obtener una estimación $\hat{P}$ de P. También se demuestra en [20] el siguiente teorema sobre el error de estimación de P:

Teorema 3.3.1. *La probabilidad de que el error de estimación de $\hat{P}$ sea a lo más e_P es*

$$Pr\left(|P - \hat{P}| \leq e_P\right) \geq 1 - 2\exp(-2^{m+1}e_p^2).$$

Así, el tamaño m nos permite ajustar el error de estimación e_P deseado. Finalmente, como P es función creciente de $\Phi(A, B)$, utilizando el método de la bisección es posible obtener una aproximación $\hat{\Phi}$ de $\Phi(A, B)$ para la cual se obtiene el valor de $\hat{P}$.

Además del error numérico de la aproximación, también hay un error en la estimación de $\Phi(A, B)$ por usar $\hat{P}$ en lugar de P. En [20] se afirma que la gráfica de P como función de $\Phi(A, B)$ muestra que en un punto α se tiene que

$$\frac{e_\Phi}{e_P} = P'(\alpha).$$

Siendo e_Φ y e_P los errores de las estimaciones $\hat{\Phi}$ y $\hat{P}$, respectivamente. Así, a través del error e_P y de la derivada se controla el error e_Φ.

3.3.2. Criterio directo

Sean A, B dos conjuntos tales que $|A| \leq |B|$ y $h \in (0, 1)$. En [24] se demostró que

$$|A| < h|B| \Longrightarrow J(A, B) < h. \tag{3.3.1}$$

Así, si dentro de una base de datos de secuencias genómicas se buscan sólo los pares que tienen un coeficiente de Jaccard asociado mayor a un umbral h, la implicancia anterior permite establecer un criterio para descartar los pares que no cumplan la condición anterior. Sin embargo, si no se cumple la hipótesis de (3.3.1), no es posible asegurar que el coeficiente de Jaccard supere el umbral. A este criterio de selección

le llamaremos **criterio directo** y ha sido utilizado previamente [24]. Este criterio se definirá formalmente en el capítulo **Métodos propuestos**.

Se destaca que este criterio no tiene margen de error en el descarte de un par: si un par de secuencias se descarta, es porque el coeficiente de Jaccard asociado es menor que h. Sin embargo tiene una importante debilidad: sólo compara qué tan similares son los tamaños de los respectivos conjuntos, sin importar la información compartida entre ellos. Así, si en una base de datos se tienen conjuntos disjuntos de tamaños suficientemente similares (de forma que se cumpla la hipótesis de (3.3.1)), el criterio directo seleccionaría todos los pares, a pesar de que a ningún par le corresponda un Jaccard superior a h. Por otro lado, si todos los conjuntos tienen un Jaccard superior a h, entonces el criterio es perfecto, pues seleccionará a todos los pares. Podemos concluir que no es posible estimar con anterioridad qué tan efectivo será el criterio al momento de descartar pares de una base de datos.

Capítulo 4

Métodos propuestos

4.1. Definición del problema

Sean $h \in (0,1)$, $k, N \in \mathbb{N}$ y $\mathcal{M} = \{L_1, L_2, L_3, .., L_N\}$ un conjunto de N secuencias genómicas. Supongamos que se obtienen todos los sketches de Hyperloglog de tamaño $\bar{m}$ de las secuencias en $\mathcal{M}$. Así, se tiene el conjunto de sketches:

$$S(\mathcal{M}) := \{S_{spec_k(L)} \ : \quad L \in \mathcal{M}\},$$

donde $S_{spec(L)}$ denota el sketch de Hyperloglog de tamaño $\bar{m}$ fijo asociado a la secuencia L. Los sketches de $S(\mathcal{M})$ serán denotados como sketches principales.

Se busca seleccionar el conjunto de pares de secuencias tales que el coeficiente de Jaccard correspondiente sea mayor al umbral h. Es decir, se busca el conjunto dado por

$$\mathcal{S}(\mathcal{M}, k, h) := \{(L, L') \in \mathcal{M}^2 \ : \quad L \neq L', \ \ J_k(L, L') \geq h\}.$$

Utilizando los sketches principales se puede estimar J_k para todos los pares de secuencias, esto requiere estimar la cardinalidad de las uniones correspondientes. Gracias a la propiedad de fusión del Hyperloglog, es posible estimar estas uniones comparando los respectivos sketches principales, sin embargo, esto requiere $N(N-1)/2$ comparaciones entre sketches de Hyperloglog de tamaño $\bar{m}$, dando un total de $\bar{m}N(N-1)/2$

comparaciones entre valores individuales. Por ejemplo, de acuerdo a Dashing [1], usando $\bar{m} = 2^{14}$ se obtiene un error de estimación menor al 1 %, y en este caso el número total de comparaciones entre registros de los sketches principales requeridas para el cómputo de todas las uniones es $2^{13}N(N-1)$.

El problema a resolver es definir criterios de selección que permitan clasificar los pares de secuencias como pertenecientes o no pertenecientes a $\mathcal{S}(\mathcal{M}, k, h)$. El objetivo es reducir la cantidad de comparaciones entre sketches principales a realizar para encontrar $\mathcal{S}(\mathcal{M}, k, h)$, o una aproximación, dada la naturaleza probabilística de los sketches. En particular, se utilizarán estrategias basadas en sketches auxiliares de Hyperloglog o SuperMinHash.

Así, lo que se busca es reducir el espacio de búsqueda de $\mathcal{M}^2$ a algún subconjunto de $\mathcal{M}^2$ de menor tamaño a través de un proceso rápido en el sentido que el tiempo de ejecución total no supere al tiempo de ejecución resultante de no utilizar ningún criterio (realizar las $N(N-1)/2$ comparaciones de sketches principales directamente). Por ejemplo, el criterio directo (Definición 4.2.1) permite descartar pares de forma directa comparando sólo dos números (las cardinalidades individuales) para reducir el espacio de búsqueda, sin embargo ya vimos que este proceso tiene un problema importante al no utilizar nada de la información compartida entre las secuencias.

El objetivo de este trabajo es reducir el tiempo en la búsqueda de pares de secuencias genómicas similares. Con este fin las secuencias se representan como multiconjuntos y se estiman las similitudes a través del coeficiente de Jaccard. Para estimar el coeficiente de Jaccard se utilizan sketches de Hyperloglog principales, considerando $\bar{m} = 2^{14}$. Así, podemos abstraernos del concepto de secuencia genómica y pensar sólo en multiconjuntos de k-mers.

El coeficiente de Jaccard entre dos multiconjuntos de k-mers A y B está dado por

$$J(A, B) = \frac{|A| + |B| - |A \cup B|}{|A \cup B|}.$$

Dado que por cada multiconjunto de k-mers se tiene su respectivo sketch de Hyperloglog principal, entonces las cardinalidades individuales se pueden estimar sin

Notación	Significado
k	Largo usado para generar las k-mers
$\mathcal{M}$	Conjunto con las secuencias genómicas
N	Cantidad de secuencias genómicas en $\mathcal{M}$
h	Umbral para la similitud
a	Cantidad de bytes adicionales que utiliza un sketch auxiliar asociado a un criterio de selección
$\bar{m}$	Tamaño de los sketches principales de Hyperloglog
m	Tamaño del sketch auxiliar de Hyperloglog ($a = m = 2^p$) o de SuperMinHash ($a = 8m$)
p	Medida asociada al tamaño del sketch auxiliar de Hyperloglog ($p = \log_2(m)$)
σ_p	Error relativo estándar de un sketch de Hyperloglog de tamaño 2^p
Z_α	Parámetros relacionado al intervalo de confianza de la estimación del Hyperloglog
n	Orden de la expansión de Taylor utilizada en el criterio hll_a de orden n
r, b	Número de filas (r) y de bandas (b) usados para la partición del criterio smh_a

Cuadro 4.1: Tabla con notaciones importantes usadas en este informe.

problemas. Por otro lado estimar $|A \cup B|$ también es directo, sin embargo esto requiere comparar dos sketches de Hyperloglog de tamaño 2^{14}, y cuando se tienen N multiconjuntos, esto corresponde a realizar $N(N-1)/2$ comparaciones entre sketches de tamaño 2^{14}. Buscamos disminuir el tiempo de búsqueda reduciendo el espacio de búsqueda, evitando así realizar todas las $N(N-1)/2$ comparaciones entre sketches principales.

Entendemos por criterio de selección a un algoritmo de decisión que recibe como entrada un par de multiconjuntos, y retorna SÍ o NO. Retorna SÍ cuando el par tiene una alta probabilidad de ser similar, y retorna NO, cuando no. Cuando el criterio retorne SI, diremos que el par fue seleccionado, y cuando retorne NO, que fue descartado. En nuestro problema, dos pares serán similares si su coeficiente de Jaccard es mayor o igual a un umbral h establecido. Así, nuestro espacio de búsqueda se reducirá a aquellos pares de multiconjuntos que fueron seleccionados por el criterio.

Este enfoque es motivado por el hecho que en bases de datos de secuencias genómicas la gran mayoría de pares son disimilares, por lo que se gasta mucho tiempo de cómputo en estimar la similitud de pares que no eran realmente similares. De este

modo, si antes de estimar la similitud real de los pares se pudiera descartar gran parte de estos pares disimilares sin perder pares similares, se podría ahorrar bastante tiempo al momento de encontrar sólo los pares similares.

4.2. Criterios de selección

A continuación se presentan cuatro criterios de selección que permiten reducir efectivamente el espacio de búsqueda. Estos son:

1. Criterio directo.
2. Criterio hll_a directo.
3. Criterio hll_a de orden n.
4. Criterio smh_a.

El primero de ellos se define en [24], y el último es una adaptación de una técnica ya existente que utiliza el sketch de SuperMinHash. Los otros dos criterios son nuevos y se definen en este trabajo. Fuera del criterio directo, todos los demás utilizan algún sketch adicional a los principales para resolver el problema, a estos sketches adicionales les llamamos sketches auxiliares.

4.2.1. Criterio directo

Este criterio ya está descrito en su totalidad en el capítulo **Marco teórico**. Por completitud, se agrega su definición formal (4.2.1) además del algoritmo correspondiente.

Definición 4.2.1 (Criterio directo)**.** *Sean A, B dos multiconjuntos tales que $|A| \leq |B|$. Entonces,*

- *Si $|A| < h|B|$, entonces el par de secuencias se descarta.*
- *Si $|A| \geq h|B|$, entonces el par de secuencias se selecciona.*

El algoritmo 3 plantea la aplicación de este criterio. En este caso el método $S.estimate()$ retorna la cardinalidad estimada por el sketch S. Como se mencionó anteriormente, el criterio retorna NO cuando decide descartar el par, y retorna SI cuando es un par candidato a ser similar.

Algoritmo 3 Criterio directo

Entrada: Par de sketches principales de Hyperloglog $\bar{S}_1, \bar{S}_2$ tales que la cardinalidad estimada de $\bar{S}_1$ es menor o igual a la cardinalidad estimada de $\bar{S}_2$; umbral $h \in (0, 1)$.

1: $|A| \leftarrow \bar{S}_1.estimate()$
2: $|B| \leftarrow \bar{S}_2.estimate()$
3: **if** $|A| < h|B|$ **then**
4: **return** NO
5: **end if**
6: **return** SI

4.2.2. Criterio hll_a de orden n

Sean A, B dos multiconjuntos no vacíos de cardinalidades fijas tales que $\gamma|A| = |B|$, con $\gamma \in [0, 1]$, es decir, $|A| \geq |B|$. Sean además $t = |A \cup B|$ y $\hat{t}_p = |A \cup B|_p$ el tamaño de la unión entre A y B, y su estimación dada por un Hyperloglog de tamaño 2^p, respectivamente. Según la propiedad sobre la estimación del sketch de Hyperloglog establecida en (2.4.2), se tiene que

$$\hat{t}_p \in I := [t(1 - Z_\alpha \sigma_p), t(1 + Z_\alpha \sigma_p)] \subset \mathbb{R}^+,$$

con una probabilidad de α.

De lo anterior se pueden deducir las siguientes desigualdades (asumiendo que $\hat{t}_p \in I$):

$$0 \leq |\hat{t}_p - t| \leq Z_\alpha \sigma_p t. \tag{4.2.1}$$

$$0 < \frac{\hat{t}_p}{1 + Z_\alpha \sigma_p} \leq t. \tag{4.2.2}$$

Luego, utilizando la desigualdad (4.2.2) y recordando que $|A| \leq t$, se tiene que

$$\frac{|A|}{t} \leq 1, \qquad \text{y} \qquad \frac{|A|}{t} \leq \frac{|A|(1 + Z_\alpha \sigma_p)}{\hat{t}_p}.$$

Así, como $(1 + \gamma) > 0$,

$$\frac{(1+\gamma)|A|}{t} \leq (1+\gamma) \operatorname{mín} \left\{ 1, \frac{|A|(1 + Z_\alpha \sigma_p)}{\hat{t}_p} \right\}. \tag{4.2.3}$$

Ahora, consideremos la función $f_J : I \to \mathbb{R}$ dada por

$$f_J(x) = \frac{|A| + |B| - x}{x} = \frac{(1+\gamma)|A|}{x} - 1.$$

Notemos que $f_J(t) = J(A, B)$. Además, $f_J(\hat{t}_p)$ es una estimación de $J(A, B)$ que denotaremos por $\hat{J}_p(A, B)$. Nuestro objetivo es estimar $f_J(t)$ utilizando $f_J(\hat{t}_p)$, de modo que una estimación de $|A \cup B|$ sea capaz de aportar información sobre el verdadero valor de $J(A, B)$. Esta relación entre $f_J(t)$ y $f_J(\hat{t}_p)$ se construye utilizando el teorema de Taylor.

Sea $i \in \mathbb{N}$, notemos que f_J es diferenciable i veces en I. En efecto,

$$\forall x \in I : \quad f_J^{(i)}(x) = (-1)^i i! \frac{(1+\gamma)|A|}{x^{i+1}}.$$

Luego, dado $x_0 \in I$, y aplicando el Teorema de Taylor, se tiene que

$$\forall x \in I : \quad f_J(x) = \sum_{i=0}^{n} \frac{f_J^{(i)}(x_0)}{i!}(x - x_0)^i + R_n(x).$$

Tomando $x = \hat{t}_p \in I$ y $x_0 = t \in I$, se tiene que

$$f_J(\hat{t}_p) = \sum_{i=0}^{n} \frac{f_J^{(i)}(t)}{i!}(\hat{t}_p - t)^i + R_n(\hat{t}_p).$$

Luego, separando el primer término de la sumatoria y recordando que $f_J(t) = J(A, B)$ y $f_J(\hat{t}_p) = \hat{J}_p(A, B)$, se tiene que:

$$\hat{J}_p(A, B) - J(A, B) \approx \sum_{i=1}^{n} (-1)^i \frac{(1+\gamma)|A|}{t^{i+1}}(\hat{t}_p - t)^i. \tag{4.2.4}$$

De esta forma, podemos acotar a $e_J^{(p)}(A,B) := |\hat{J}_p(A,B) - J(A,B)|$ como sigue:

$$\begin{aligned}
|\hat{J}_p(A,B) - J(A,B)| &\approx \left| \sum_{i=1}^{n} (-1)^i \frac{(1+\gamma)|A|}{t^{i+1}} (\hat{t}_p - t)^i \right| \\
&\leq \sum_{i=1}^{n} \left| (-1)^i \frac{(1+\gamma)|A|}{t^{i+1}} (\hat{t}_p - t)^i \right| \\
&= \sum_{i=1}^{n} \frac{(1+\gamma)|A|}{t^{i+1}} |\hat{t}_p - t|^i \\
&\leq \sum_{i=1}^{n} \frac{(1+\gamma)|A|}{t^{i+1}} t^i Z_\alpha^i \sigma_p^i \qquad \text{(Por (4.2.1))} \\
&= \frac{(1+\gamma)|A|}{t} \sum_{i=1}^{n} (Z_\alpha \sigma_p)^i
\end{aligned}$$

Finalmente, utilizando (4.2.3), se tiene que

$$|\hat{J}_p(A,B) - J(A,B)| \leq (1+\gamma) \operatorname{mín} \left\{ 1, \frac{|A|(1+Z_\alpha\sigma_p)}{\hat{t}_p} \right\} \sum_{i=1}^{n} (Z_\alpha \sigma_p)^i.$$

Utilizar el valor absoluto se justifica por la desigualdad triangular inversa (DTI). En efecto, si M_n denota a la sumatoria del miembro derecho de (4.2.4), entonces

$$\begin{aligned}
\Big| |\hat{J}_p(A,B) - J(A,B)| - |M_n| \Big| &= \Big| |\hat{J}_p(A,B) - J(A,B)| - |-M_n| \Big| \\
&\leq \Big| \Big(\hat{J}_p(A,B) - J(A,B) \Big) + (-M_n) \Big| \qquad \text{(DTI)} \\
&= |R_n(\hat{t}_p)|.
\end{aligned}$$

Así, si $\hat{J}_p(A,B) - J(A,B)$ es cercano a M_n, entonces $|\hat{J}_p(A,B) - J(A,B)|$ es igual de cercano a $|M_n|$ (o más cercano).

Luego, definiendo

$$C_{p,\alpha,n}(A,B) := (1+\gamma) \operatorname{mín} \left\{ 1, \frac{|A|(1+Z_\alpha\sigma_p)}{\hat{t}_p} \right\} \sum_{i=1}^{n} (Z_\alpha \sigma_p)^i,$$

se tiene que

$$e_J^{(p)}(A,B) \leq C_{p,\alpha,n}(A,B). \tag{4.2.5}$$

Hemos encontrado entonces una cota para $e_J^{(p)}(A,B)$. Esta cota nos dice que la distancia entre $J(A,B)$ y $\hat{J}_p(A,B)$ es a lo más de $C_{p,\alpha,n}(A,B)$. Además,

$$C_{p,\alpha,n}(A,B) \leq (1+\gamma)\sum_{i=1}^{n}(Z_\alpha \sigma_p)^i,$$

obteniéndose una cota independiente de la estimación $\hat{t}_p$. Esta puede ser útil para calibrar el tamaño del Hyperloglog a usar antes de realizar las comparaciones.

Es importante que $\hat{t}_p$ pertenezca a I para que así las desigualdades se cumplan. Por esta razón se utilizará $\alpha = 95\,\%$ a menos que se especifique lo contrario. De esta forma $Z_\alpha = 1.96$. También sería válido tomar $Z_\alpha = 3$ para tener una confianza del $99.73\,\%$, sin embargo es necesario que Z_α sea lo menor posible para que $C_{p,\alpha,n}$ sea lo más ajustada posible.

Utilizando la misma notación definida en esta subsección, notemos que

$$|J(A,B) - \hat{J}_p(A,B)| \leq C_{p,\alpha,n}(A,B) \Longrightarrow J(A,B) - \hat{J}_p(A,B) \leq C_{p,\alpha,n}.$$

Así, $J(A,B) \leq \hat{J}_p(A,B) + C_{p,\alpha,n}(A,B)$, por lo que

$$\hat{J}_p(A,B) + C_{p,\alpha,n}(A,B) < h \Longrightarrow J(A,B) < h. \tag{4.2.6}$$

De esta forma es posible utilizar el sketch auxiliar de tamaño m para clasificar el par (A,B) como similar o no, además de los sketches principales de tamaño $\bar{m}$. Consideremos el parámetro a como la memoria,en bytes, utilizada por el sketch de Hyperloglog auxiliar. En la implementación, el sketch utiliza un arreglo de m registros de 8 bits, es decir, el sketch auxiliar utilizaría m bytes, así, $a = m$ en este caso.

Con todo lo anterior podemos definir el criterio hll_a de orden n como sigue.

Definición 4.2.2 (Criterio hll_a de orden n)**.** *Sea $n \in \mathbb{N}$ y sean A, B dos multiconjuntos. Si $p = \log_2(a)$ y siguiendo la notación ya establecida anteriormente, entonces:*

- *Si $\hat{J}_p(A,B) + C_{p,\alpha,n} < h$, entonces el par de secuencias se descarta.*

- *Si* $\hat{J}_p(A,B) + C_{p,\alpha,n} \geq h$*, entonces el par se selecciona.*

En la Figura 4.1 se muestra un esquema que ejemplifica el funcionamiento del criterio propuesto.

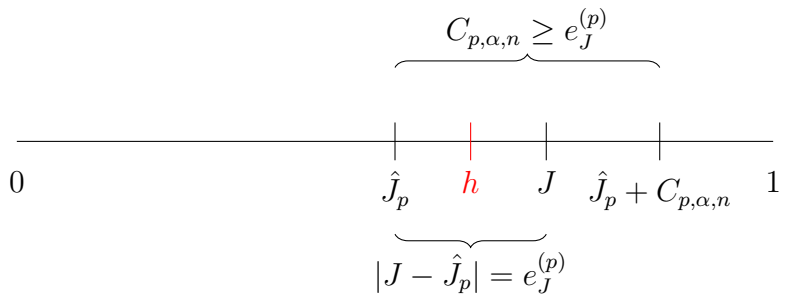


Figura 4.1: Esquema criterio hll_a. $C_{p,\alpha,n}$ permite construir un intervalo centrado en $\hat{J}_p$ donde se espera que se encuentre J. Así, si el extremo derecho de este intervalo $\hat{J}_p + C_{p,\alpha,n}$ es menor que h, entonces se espera que J también sea menor que h.

Es claro entonces que el nuevo criterio depende principalmente de dos cosas. Primero, que la cota propuesta $C_{p,\alpha,n}$ se cumpla. De esta forma podemos asegurar que $J \leq \hat{J}_p + C_{p,\alpha,n}$ evitando que hayan falsos negativos. Lo segundo es que la cota propuesta $C_{p,\alpha,n}$ sea lo más pequeña posible, de esta forma $\hat{J}_p + C_{p,\alpha,n}$ está mucho más cerca de J, resultando en menos falsos positivos.

El algoritmo 4 describe el funcionamiento de este criterio. En este caso el método $S.estimate()$ retorna la cardinalidad estimada por el sketch S, y el método $S_1.union\_size(S_2)$ retorna la cardinalidad estimada por los sketches S_1 y S_2 de la respectiva unión. El criterio retorna NO cuando decide descartar el par, y retorna SI cuando es un par candidato a ser similar.

Algoritmo 4 Criterio hll_a de orden n

Entrada: Par de sketches principales de Hyperloglog $\bar{S}_1, \bar{S}_2$; par de sketches auxiliares de Hyperloglog S_1, S_2 de tamaño $m = a$ asociados a S_1 y S_2 (mismos conjuntos procesados), respectivamente; $n \in \mathbb{N}$; umbral $h \in (0, 1)$.

1: $|A| \leftarrow \bar{S}_1.estimate()$
2: $|B| \leftarrow \bar{S}_2.estimate()$
3: $\hat{t}_p \leftarrow S_1.union\_size(S_2)$
4: $\hat{J} \leftarrow \dfrac{|A| + |B| - \hat{t}_p}{\hat{t}_p}$.
5: $C \leftarrow C_{p,\alpha,n}$
6: **if** $\hat{J}_p(A, B) + C_{p,\alpha,n} < h$ **then**
7: **return** NO
8: **end if**
9: **return** SI

4.2.3. Criterio hll_a directo

Sean A, B dos multiconjuntos. De la deducción del criterio directo, y siguiendo las mismas notaciones, se desprende también que

$$\begin{aligned} J(A,B) &= f_J(t) \\ &= \frac{|A| + |B| - t}{t} \\ &\leq \frac{|A| + |B| - \hat{t}_p^+}{\hat{t}_p^+} \qquad \text{(De la desigualdad (4.2.2).)} \\ &=: K_{p,\alpha}^+(A,B), \end{aligned}$$

donde $\hat{t}_p^+ = \dfrac{\hat{t}_p}{1 + Z_\alpha \sigma_p}$. De este modo, se espera que $J(A, B) \leq K_{p,\alpha}^+(A, B)$. Esto permite definir el siguiente criterio de selección:

Definición 4.2.3 (Criterio hll_a directo)**.** *Sean A, B dos multiconjuntos. Si $p = \log_2(a)$ y siguiendo la notación establecida anteriormente, entonces*

- *Si $K_{p,\alpha}^+(A, B) < h$, entonces el par de secuencias se descarta.*
- *Si $K_{p,\alpha}^+(A, B) \geq h$, entonces el par de secuencias se selecciona.*

El algoritmo 5 describe el funcionamiento del criterio directo. En este caso el método $S.estimate()$ retorna la cardinalidad estimada por el sketch S, y el método $S_1.union\_size(S_2)$ retorna la cardinalidad estimada por los sketches S_1 y S_2 de la respectiva unión. El criterio retorna NO cuando decide descartar el par, y retorna SI cuando es un par candidato a ser similar.

Algoritmo 5 Criterio hll_a directo

Entrada: Par de sketches principales de Hyperloglog $\bar{S}_1, \bar{S}_2$; par de sketches auxiliares de Hyperloglog S_1, S_2 de tamaño $m = a$ asociados a S_1 y S_2, respectivamente; umbral $h \in (0, 1)$.

1: $|A| \leftarrow \bar{S}_1.estimate()$
2: $|B| \leftarrow \bar{S}_2.estimate()$
3: $\hat{t}_p \leftarrow S_1.union\_size(S_2)$
4: $K \leftarrow K^+_{p,\alpha}(A, B)$.
5: **if** $K < h$ **then**
6: **return** NO
7: **end if**
8: **return** SI

4.2.4. Criterio smh_a

La técnica LSH que utiliza sketches SuperMinHash ya se presentó en el capítulo **2**. Ahora sólo se adaptarán los parámetros y conceptos para definir el criterio de selección como tal.

Dado que se quiere retener la mayor cantidad de pares similares, se opta por establecer una configuración de parámetros b y r tales que

$$0.95 \leq 1 - (1 - h^r)^b, \tag{4.2.7}$$

de esta forma, los pares cuyo coeficiente de Jaccard sean mayores que h serán seleccionados con una probabilidad de al menos un 95 %. Para encontrar $r, b \in \mathbb{N}$ que satisfagan esto, basta notar que $r = m/b$, por lo que la desigualdad (4.2.7) se

transforma en

$$0.95 \leq 1 - (1 - h^{\frac{m}{b}})^b,$$

donde m y h son parámetros preseleccionados. Luego, como la función $b \mapsto 1 - (1 - h^{\frac{m}{b}})^b$ es creciente en b (ver **Apéndice**, demostración 6.2.1), basta ir revisando los divisores de m en orden creciente hasta encontrar el primero valor de b que satisfaga esta desigualdad, y r se calcula a partir de la igualdad $rb = m$. Se utiliza el primer valor que funcione porque si bien queremos seleccionar a la mayor cantidad de pares similares, también queremos que este criterio descarte la mayor cantidad de pares disimilares.

Similar a los criterios anteriores, consideremos el parámetro a como la cantidad de bytes que se necesitan para almacenar un sketch auxiliar de SuperMinHash de tamaño m. Un sketch de SuperMinHash se implementa utilizando m registros de 8 bytes cada uno, resultando en $8m$ de bytes en total. Así, en este caso $a = 8m$.

Con todo lo anterior ya podemos definir el criterio smh_a.

Definición 4.2.4 (criterio smh_a)**.** *Sean A, B dos multiconjunto, y sea $m = a/8$. Supongamos que $S_m(A)$ y $S_m(B)$ son los sketches de SuperMinHash de tamaño m asociados a A y B, respectivamente. Supongamos además que r y b son los valores encontrados para particionar los sketches de SuperMinHash según lo descrito anteriormente. Entonces*

- *Si existe una banda para la cual $S_m(A)$ y $S_m(B)$ son iguales, entonces se selecciona el par.*
- *Si para todas las bandas se tiene que $S_m(A)$ y $S_m(B)$ no coinciden, entonces el par se descarta.*

El algoritmo 6 describe el funcionamiento del criterio smh_a. La notación $S[i]$ corresponde al elemento en la posición i-ésima del sketch S (contando desde el índice 0). Así, el primer ciclo *for* recorre las bandas del sketch, y el segundo ciclo *for* itera sobre las filas correspondientes a esa banda, si alguna de las filas es diferente, pasamos a la siguiente banda, en caso contrario, todas las filas son iguales y por lo tanto el

criterio retorna SI (se selecciona el par). Si ninguna banda coincide, se retorna NO y se descarta el par.

El criterio retorna NO cuando decide descartar el par, y retorna SI cuando es un par candidato a ser similar.

Algoritmo 6 Criterio smh_a

Entrada: Par de sketches par de sketches auxiliares de SuperMinHash S_1, S_2 de tamaño $m = a/8$; número de filas r; número de bandas b; umbral $h \in (0, 1)$.

```
for i = 0, 1, 2, ..., b − 1 do
    equal_block ← TRUE
    for j = 0, 1, 2, ..., r − 1 do
        if S1[i · r + j] ≠ S2[i · r + j] then
            equal_block ← FALSE;
        end if
    end for
    if equal_block then
        return SI
    end if
end for
return NO
```

4.3. Criterios inversos

De la deducción del criterio hll_a de orden n, notemos que la cota $C_{p,\alpha,n}$ también permite establecer la siguiente implicancia:

$$|J(A,B) - \hat{J}_p(A,B)| \leq C_{p,\alpha,n}(A,B) \Longrightarrow \hat{J}_p(A,B) - C_{p,\alpha,n} \leq J(A,B).$$

Así,

$$h < \hat{J}_p(A,B) - C_{p,\alpha,n} \Longrightarrow h < J(A,B).$$

Esta implicancia permite establecer un criterio de selección para encontrar los pares cuyo coeficiente de Jaccard es a lo más h, o dicho de otra forma, permite identificar los pares disimilares. Formalmente, este criterio se define como sigue:

Definición 4.3.1 (Criterio hll_a de orden n inverso)**.** *Sea $n \in \mathbb{N}$ y sean A, B dos multiconjuntos. Si $p = \log_2(a)$ y siguiendo la notación ya establecida. Entonces*

- *Si $\hat{J}_p(A,B) - C_{p,\alpha,n} < h$, entonces el par de secuencias se descarta.*
- *Si $\hat{J}_p(A,B) - C_{p,\alpha,n} \leq h$, entonces el par se selecciona.*

Sucede algo parecido con el criterio hll_a directo. De (2.4.2) también es posible deducir que

$$t \geq \hat{t}_p^- := \frac{\hat{t}_p}{1 - Z_\alpha \sigma_p},$$

siempre que $1 - Z_\alpha \sigma_p \geq 0$, lo cual es cierto para p suficientemente grande. Luego, se obtiene que

$$J(A,B) = \frac{|A| + |B| - t}{t} \geq K_{p,\alpha}^-(A,B),$$

donde $K_{p,\alpha}^-(A,B) := \dfrac{|A| + |B| - \hat{t}_p^-}{\hat{t}_p^-}$. Este hecho se formaliza en el siguiente criterio de selección:

Definición 4.3.2 (Criterio hll_a directo inverso)**.** *Sean A, B dos multiconjuntos. Si $p = \log_2(a)$ y siguiendo la notación ya establecida. Entonces*

- *Si $K_{p,\alpha}^-(A,B) \geq h$, entonces el par de secuencias se descarta.*
- *Si $K_{p,\alpha}^-(A,B) < h$, entonces el par de secuencias se selecciona.*

Los criterios de selección inversos, que llamamos así porque buscan pares disimilares, no serán evaluados experimentalmente, pues no tienen alguna utilidad práctica en el contexto del problema que se está intentando resolver con las bases de datos de secuencias genómicas. Sin embargo es interesante esta opción alternativa que se desprende de los criterios basados en Hyperloglog y su análisis y potencial impacto para algunas aplicaciones se puede realizar en un trabajo futuro.

Capítulo 5

Evaluación experimental

Es necesario poner a prueba los distintos criterios de selección, para ello se evaluarán principalmente dos aspectos: la clasificación de los pares (similar o no) y el tiempo que se ahorra al integrar el respectivo criterio de selección. En esta capítulo se presentan los resultados experimentales que se obtienen al resolver el problema de encontrar los pares similares. Primero se detallan los datos y las métricas que serán utilizadas, y luego, los resultados en distintos escenarios.

5.1. Métricas de desempeño

Los criterios de selección, al ser algoritmos de clasificación, permiten usar las definiciones típicas de datos clasificados, que en este caso son pares de secuencias:

- **Verdadero Positivo o True Positive (*TP*)**: Par cuyo coeficiente de Jaccard asociado es mayor o igual que h y es seleccionado por el criterio.
- **Falso Positivo o False Positive (*FP*)**: Par cuyo coeficiente de Jaccard asociado es menor que h y es seleccionado por el criterio.
- **Verdadero Negativo o True Negative (*TN*)**: Par cuyo coeficiente de Jaccard asociado es menor que h y es descartado por el criterio.

- **Falso Negativo o False Negative (*FN*)**: Par cuyo coeficiente de Jaccard asociado es mayor o igual que h y es descartado por el criterio.

El objetivo de los criterios de selección propuestos es descartar la mayor cantidad de pares de secuencias disimilares mientras seleccionan la mayor cantidad de pares similares. Así, las dos métricas que se usarán para medir el desempeño en la clasificación de pares son la sensibilidad y la especificidad. La sensibilidad y la especificidad, que denotaremos por TPR (True Positive Rate) y TNR (True Negative Rate), respectivamente, están dadas por:

$$TPR = \frac{TP}{TP + FN}, \qquad \text{(Sensibilidad)}$$

$$TNR = \frac{TN}{TN + FP}. \qquad \text{(Especificidad)}$$

El TPR se entiende como la porción de pares similares que fueron seleccionados por el criterio, a su vez, el TNR corresponde a la porción de pares disimilares que fueron descartados por el criterio.

Así, el TPR y el TNR permiten estudiar qué tan bien clasifican los criterios a los pares de secuencias. Se requiere que el TPR sea muy cercano a 1, pues los pares similares corresponde a la información valiosa que se quiere obtener. Por otro lado, el TNR se desea sea lo más alto posible, ojalá cercano a 1, pues mientras más pares negativos se descarten, más se reduce el espacio de búsqueda. Se espera entonces que un alto TNR reduzca el tiempo de búsqueda de los pares similares.

También es necesario estudiar si efectivamente el tiempo de ejecución. Para analizar este aspecto se utiliza la aceleración. La aceleración de un experimento A con respecto a un experimento B está dada por:

$$\text{Aceleración} = \frac{\text{Tiempo experimento } B}{\text{Tiempo experimento } A}.$$

Así, una aceleración mayor a 1 indica que el experimento A tarda menos tiempo que el experimento B. De esta forma, podemos comparar cuántas veces más rápido es

utilizar un criterio de selección con respecto a no utilizar ningún criterio al buscar los pares similares.

5.2. Datos a utilizar

Conjuntos de secuencias genómicas de la base de datos de RefSeq fueron utilizados para estudiar el rendimiento de los criterios de selección establecidos, estos se muestran en el cuadro 5.1. El nombre indica el origen de las secuencias genómicas, en particular, *prefs* corresponde a un conjunto de secuencias de orígenes variados, el cual fue construido por los autores de Dashing [1] con el fin de abarcar pares cuyos coeficientes de Jaccard se distribuyan en todo el rango $[0, 1]$. Además, *balanced* es un conjunto de cien mil pares de bacterias tales que diez mil de ellos tienen un coeficiente de Jaccard entre $0.1i$ e $(i + 1)0.1$, con $i \in \{0, 1, 2, ..., 9\}$. De esta forma, *balanced* busca representar una situación en que los coeficientes de Jaccard se distribuyan balanceadamente en el rango $[0, 1]$.

La columna "Símbolo" corresponde al símbolo con el que se identificará al dataset, la columna "# sec." representa la cantidad de secuencias en el conjunto, la siguiente columna "# pares" es la cantidad de pares que se pueden formar con elementos del conjunto, es decir, corresponde a la cantidad de comparaciones entre sketches principales para poder estimar todas las uniones. Las últimas dos columnas hacen referencias al tamaño máximo y mínimo en memoria entre las secuencias de cada dataset.

El tamaño en memoria entregado en las últimas dos columnas es proporcional al largo de las secuencias. Así, este tamaño permite observar que los distintos conjuntos abarcan distintos largos de secuencias, desde conjuntos con secuencias de tamaño del orden de 0.1 KB, hasta conjuntos con secuencias del tamaño del orden de 10 GB. De esta forma podemos evaluar los criterios en variados contextos de tamaños de secuencias.

Conjunto	Símbolo	# sec.	# pares	Tamaño min.	Tamaño max.
protozoa	$\mathcal{U}_1$	92	4186	6.3MB	222.0MB
plant	$\mathcal{U}_2$	145	10440	13.0MB	11.0GB
vertebrate mammalian	$\mathcal{U}_3$	187	17391	129.0MB	4.2GB
vertebrate other	$\mathcal{U}_4$	271	36585	69.0MB	5.1GB
prefs	$\mathcal{U}_5$	281	39340	5.4MB	12.0MB
fungi	$\mathcal{U}_6$	313	48828	2.2MB	133.0MB
archaea	$\mathcal{U}_7$	692	239086	480.0KB	5.6MB
viral	$\mathcal{U}_8$	11232	63073296	0.4KB	2.4MB
bacteria50k	$\mathcal{U}_9$	50000	1249975000	111.0KB	4.1MB

Cuadro 5.1: Detalle de los conjuntos con los que se realizó la experimentación.

5.3. Algoritmo general e implementación de los criterios de selección

El procedimiento general se describe en el pseudo código del algoritmo 7. Si bien la implementación computacional incluye la construcción de los diferentes sketches, el problema que estamos abordando es cómo acelerar el proceso de comparaciones de similitud utilizando memoria adicional. Estas comparaciones se realizan en el doble ciclo **for**, donde un criterio de selección cualquiera decide si el par es seleccionado o no en base a los parámetros y entradas correspondientes ($criterioParams$ y h), en caso de ser seleccionado, se calcula su coeficiente de Jaccard y se termina de revisar si es que el respectivo par es similar o no según el umbral h. De esta forma, la función *criterio* que aparece en la línea 5 se reemplazará por cada uno de los criterios de selección y se evaluará el tiempo de comparación como el tiempo que tarda en ejecutarse el doble ciclo **for**.

Se utilizó la biblioteca disponible en GitHub (`https://github.com/dnbaker/sketch`) de Daniel Baker para procesar las secuencias y construir los sketches de Hyperloglog principales, además de los sketches auxiliares de Hyperloglog para la aplicación del criterio hll_a directo y del criterio hll_a de orden n. Esta biblioteca fue desarrollada por los autores de Dashing [1].

Algoritmo 7 Algoritmo general

Entrada: Conjunto de N secuencias genómicas $\mathcal{M}$ y un umbral $h \in (0, 1)$.

1: Generar los sketches principales $S_1, S_2, ..., S_N$ de las N secuencias.
2: Generar los sketches auxiliares $S'_1, S'_2, ..., S'_N$ de las N secuencias.*
3: **for** $i = 1, 2, ..., N - 1$ **do**
4: **for** $j = i + 1, i + 2, ..., N$ **do**
5: $seleccion \leftarrow criterio(criterioParams, h)$;
6: **if** not *seleccion* **then**
7: **continue**;
8: **end if**
9: $J \leftarrow Jaccard(S_i, S_j)$;
10: **if** $J \geq h$ **then**
11: Seleccionar par $(\mathcal{M}[i], \mathcal{M}[j])$;
12: **end if**
13: **end for**
14: **end for**

* Esta línea no se ejecuta en caso de estar utilizando el criterio directo.

En el caso del criterio smh_a, la misma biblioteca *sketch* ofrece una implementación de SuperMinHash [8], una versión refinada de MinHash. Esta implementación de SuperMinHash es la que se utiliza en este trabajo para la construcción de los sketches de MinHash, ya que como menciona su autor Otmar Ertl, SuperMinHash funciona como un reemplazo de MinHash, dado que mantiene las mismas propiedades que MinHash o incluso las mejora. En particular, la varianza del estimador del coeficiente de Jaccard se reduce por un factor $\alpha \in (0, 1)$ que depende de la configuración y los datos. Teniendo los sketches de SuperMinHash, se implementa el criterio smh_a según lo descrito en el capítulo **Métodos propuestos**.

5.4. Configuración para los experimentos

Los experimentos se implementaron en C++14 y se ejecutaron en un servidor con sistema operativo Linux del departamento de Ingeniería Eléctrica de la Universidad de Concepción. Este cuenta con un procesador Intel Xeon Gold 5118 con 12 núcleos físicos (24 hebras), y 128 GB de RAM. Además, el servidor cuenta con un disco SSD de 1TB NVMe Class.

Para compilar se utilizó `g++` (GCC) versión 12.2.1. con las opciones `-std=c++14, -fopenmp, -O3, -match=native`. Como indica la segunda opción, se utilizó OpenMP para paralelizar el programa. En los experimentos en que se medía el tiempo de ejecución, se utilizaron 8 hebras, a excepción del caso cuando se evaluaba el conjunto *prefs* con distintos valores del umbral h, donde se utilizó sólo 1 ya que no tardaba lo suficiente como para justificar la paralelización. En cuanto a los experimento en que se medían las métricas de clasificación, se utilizaban en general más de 10 hebras, el resultado en este caso es independiente de la cantidad de hebras, estas sólo reducen el tiempo de cálculo de las métricas de clasificación.

5.5. Resultados

Para medir el rendimiento de los distintos criterios, primero se evaluarán utilizando las métricas de clasificación *TPR* y *TNR*. Luego, se comparará el impacto de aplicar los criterios, tanto en el espacio en memoria como en tiempo, para resolver el problema de encontrar los pares de secuencias similares.

5.5.1. Desempeño para $h = 0.8$

A continuación se presentarán los resultados para las métricas de desempeño propuestas para los distintos criterios y conjuntos de datos, considerando un umbral constante de $h = 0.8$. Una similitud de Jaccard de 0.8 corresponde a un ANI reescalado de 0.9962, según la relación con la distancia Mash dada por

$$ANI_{est} = 1 + \frac{1}{k} \ln \left(\frac{2J}{1+J} \right).$$

Primero se mostrarán las métricas de clasificación *TPR* y *TNR*, a modo de estudiar el comportamiento de los distintos criterios. Recordemos que es necesario que el *TPR* sea muy cercano a 100 %, pues no queremos descartar la información valiosa: los pares similares. Además, un alto *TNR* es necesario para que el criterio acelere el proceso de búsqueda de los pares similares, pues gasta menos tiempo revisando los pares que no lo son.

Criterio directo

El primer criterio a evaluar es el criterio directo. El cuadro 5.2 muestra las métricas de clasificación, en porcentaje, que se obtienen al utilizar este criterio. Como se mencionó anteriormente, la gran ventaja de este criterio es que todos los pares similares son correctamente seleccionados, y esto se refleja en el *TPR* del 100 % obtenido en todos los casos. Nunca se pierde un par de secuencias similares. Sin embargo, el criterio no se muestra consistente al momento de descartar pares negativos, y es que el *TNR* varía desde un 22.6 % hasta un 89.9 % en los distintos conjuntos de datos. La razón de esto es que el criterio depende de qué tan diferentes sean las cardinalidades de los multiconjuntos de k-mers, por lo que todos los pares que tengan cardinalidad similar serán seleccionado, incluso si estos multiconjuntos son disjuntos. Así, qué tan bien filtre los pares negativos depende totalmente de qué tan diferente sean las cardinalidades de los multiconjuntos no similares.

Criterio hll_a de orden 1

Los cuadros 5.3-5.5, muestran los resultados del criterios hll_a de orden 1. Como es estableció en su definición, este criterio depende de qué tan efectiva sea la cota $C_{p,\alpha,n}$ propuesta, el cuadro 5.3 muestra el porcentaje de éxito de la cota, es decir, el porcentaje de pares para los cuales se cumple la desigualdad (4.2.5). Podemos observar que a lo largo de todos los conjuntos de datos y los distintos valores de

Conjunto	*TPR*	*TNR*
$\mathcal{U}_1$	100	81.1
$\mathcal{U}_2$	100	84.6
$\mathcal{U}_3$	100	48.5
$\mathcal{U}_4$	100	78.8
$\mathcal{U}_5$	100	22.6
$\mathcal{U}_6$	100	70.3
$\mathcal{U}_7$	100	67.2
$\mathcal{U}_8$	100	89.4
$\mathcal{U}_9$	100	62.6

Cuadro 5.2: Métricas de clasificación para el criterio directo.

$p = \log_2(a)$, la desigualdad se cumple, en promedio, en más del 95 % de los pares, por lo que los resultados experimentales respaldan la deducción teórica.

Con respecto a las métricas de clasificación, el cuadro 5.4 indica que el criterio selecciona casi perfectamente todos los pares positivos, se pierde, en general, menos del 1 % de la información valiosa. El único caso donde el *TPR* baja drásticamente (para el conjunto $\mathcal{U}_1$), ocurre porque habían 5 pares positivos y el criterios clasificó erróneamente a uno de ellos. Por otro lado, el cuadro 5.5 permite concluir que para valores de a mayores o iguales a 64 ($p \geq 6$) se obtiene un *TNR* cercano al 100 % en casi todas las clases, es decir, el criterio descartará correctamente a casi todos los pares negativos. Además, valores de a menores que 256 resultan en un uso de menos del 1 % de memoria adicional. Si bien valores de a más grandes resultan en un mejor filtrado sobre los pares negativos, también estaremos utilizando sketches de tamaño cercanos a los sketches principales (2^{14} bytes), por lo que calcular $\hat{J}_p$ será casi tan costoso como estimar J, resultando en un criterio que consuma más tiempo, afectando negativamente el rendimiento en general. Se espera entonces que las mejores opciones sean utilizar los valores de a más pequeños que resulten en un *TNR* lo más alto posible, como sucede con a igual $64, 128$ u 256.

Si bien el criterio puede aplicarse utilizando $n \in \mathbb{N}$, los resultados son suficientemente

Conjunto	a (memoria adicional en bytes)				
	32	128	512	2048	8192
$\mathcal{U}_1$	97.55	96.35	93.77	96.80	98.73
$\mathcal{U}_2$	97.44	97.93	93.81	95.88	98.88
$\mathcal{U}_3$	97.95	96.98	96.80	97.19	99.19
$\mathcal{U}_4$	97.28	98.09	95.87	96.05	98.87
$\mathcal{U}_5$	97.84	96.21	93.96	96.23	99.20
$\mathcal{U}_6$	96.97	97.99	96.82	97.64	99.57
$\mathcal{U}_7$	94.38	95.88	94.43	97.06	98.64
$\mathcal{U}_8$	97.81	98.19	97.63	96.89	98.82
$\mathcal{U}_9$	96.87	96.80	96.57	97.19	99.41

Cuadro 5.3: Porcentaje de éxito de la desigualdad (4.2.5).

Conjunto	a (memoria adicional en bytes)									
	16	32	64	128	256	512	1024	2048	4096	8192
$\mathcal{U}_1$	100.0	80.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_2$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_3$	90.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_4$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_5$	99.7	100.0	100.0	100.0	100.0	99.7	100.0	100.0	97.5	100.0
$\mathcal{U}_6$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_7$	100.0	100.0	98.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_8$	99.5	99.0	99.3	99.8	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_9$	99.8	99.7	99.9	99.9	100.0	100.0	100.0	100.0	100.0	100.0

Cuadro 5.4: *TPR* del criterio hll_a de orden 1.

Conjunto	a (memoria adicional en bytes)									
	16	32	64	128	256	512	1024	2048	4096	8192
$\mathcal{U}_1$	52.2	89.9	99.6	99.9	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_2$	64.1	92.8	99.6	99.9	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_3$	63.0	87.1	98.7	99.7	99.8	99.9	99.9	99.9	100.0	100.0
$\mathcal{U}_4$	58.6	91.7	99.5	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_5$	58.5	84.7	93.5	95.8	97.8	98.9	98.9	99.1	99.6	99.9
$\mathcal{U}_6$	53.4	89.5	99.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_7$	48.1	82.7	98.6	99.2	99.2	99.3	99.3	99.7	99.7	99.9
$\mathcal{U}_8$	67.2	93.0	99.8	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_9$	52.3	85.9	96.8	97.7	99.1	99.9	99.9	100.0	100.0	100.0

Cuadro 5.5: *TNR* del criterio hll_a de orden 1.

satisfactorios utilizando $n = 1$. Esto se debe principalmente a que la función f_J se puede aproximar bastante bien a través de su polinomio de Taylor gracias a que sus derivadas toman valores bastantes pequeños. Esta es la razón de que sólo se incluyan resultados para $n = 1$. (ver **Apéndice**, demostración 6.2.2)

Criterio hll_a directo

El criterio hll_a directo alcanza resultados muy similares a los del criterio hll_a de orden 1. En el cuadro 5.6 vemos que su *TPR* también es muy cercano al 100 % en todos los casos. Aunque el *TPR* es siempre menor o igual que *TPR* del criterio hll_a de orden 1, la diferencia es muy pequeña. En cuanto al *TNR*, el cuadro 5.7 muestra que este es mayor que 90 % en casi todos los casos para valores de $a \geq 32$ ($p \geq 5$). Pareciera que, en general, el filtrado es más preciso sobre los pares negativos, pero pierde un poco de precisión sobre los pares positivos.

Criterio smh_a

El criterio smh_a utiliza al sketch de SuperMinHash como sketch auxiliar. Dado que los valores de r, b deben ser divisores de m, se utilizaron valores de m que sean altamente compuestos. Si bien los tamaños no son similares a la de lo sketches de Hyperloglog, igual se pueden comparar en términos de a: bytes de memoria adicional

Conjunto	a (memoria adicional en bytes)									
	16	32	64	128	256	512	1024	2048	4096	8192
$\mathcal{U}_1$	100.0	80.0	80.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_2$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_3$	90.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_4$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_5$	99.2	100.0	100.0	100.0	99.7	98.6	100.0	100.0	97.0	100.0
$\mathcal{U}_6$	66.7	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_7$	99.8	99.7	98.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_8$	97.1	98.3	98.8	99.3	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_9$	99.3	99.1	99.7	99.8	99.9	100.0	100.0	100.0	100.0	100.0

Cuadro 5.6: *TPR* del criterio hll_a directo.

Conjunto	a (memoria adicional en bytes)									
	16	32	64	128	256	512	1024	2048	4096	8192
$\mathcal{U}_1$	71.1	94.1	99.7	99.9	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_2$	78.0	96.1	99.8	99.9	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_3$	78.1	91.2	99.3	99.8	99.8	99.9	99.9	99.9	100.0	100.0
$\mathcal{U}_4$	73.4	95.1	99.7	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_5$	80.4	91.2	94.0	96.8	98.3	99.0	98.9	99.1	99.6	99.9
$\mathcal{U}_6$	73.6	95.0	99.8	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_7$	72.8	91.2	98.9	99.2	99.3	99.3	99.3	99.7	99.8	99.9
$\mathcal{U}_8$	75.2	94.2	99.8	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_9$	75.8	93.1	97.3	98.1	99.3	99.9	99.9	100.0	100.0	100.0

Cuadro 5.7: *TNR* del criterio hll_a directo.

utilizada. El cuadro 5.8 muestra el *TPR* de este criterio, se ve que para todas las configuraciones utilizadas siempre se pueden encontrar parámetros r, b tales que el criterio seleccione casi al 100 % de los pares similares. Por otro lado, el cuadro 5.9 muestra que el *TNR* es casi del 100 % para casi todo los casos cuando $a \geq 48$. Notar que con $a = 48$ se consiguen resultados similarmente buenos en comparación a los criterios de Hyperloglog que utilizan $a = 64$ o $a = 32$.

Conjunto	a (memoria adicional en bytes)								
	16	32	48	96	192	288	384	480	960
$\mathcal{U}_1$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_2$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_3$	100.0	100.0	90.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_4$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_5$	99.7	100.0	99.7	100.0	98.6	100.0	100.0	99.4	100.0
$\mathcal{U}_6$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_7$	100.0	100.0	100.0	100.0	99.8	100.0	100.0	98.8	100.0
$\mathcal{U}_8$	100.0	100.0	99.5	99.8	99.3	100.0	100.0	99.8	100.0
$\mathcal{U}_9$	100.0	100.0	99.8	100.0	99.9	100.0	100.0	99.9	100.0

Cuadro 5.8: *TPR* del criterio smh_a.

Conjunto	a (memoria adicional en bytes)								
	16	32	48	96	192	288	384	480	960
$\mathcal{U}_1$	31.9	99.4	99.9	99.9	99.9	100.0	100.0	100.0	100.0
$\mathcal{U}_2$	20.8	95.3	99.9	99.8	100.0	99.9	99.9	100.0	99.9
$\mathcal{U}_3$	15.4	81.2	99.3	99.0	99.8	99.7	99.7	99.9	99.8
$\mathcal{U}_4$	11.1	92.3	99.9	99.8	99.9	100.0	100.0	100.0	100.0
$\mathcal{U}_5$	11.2	91.9	97.2	96.0	98.5	97.4	97.2	98.6	97.3
$\mathcal{U}_6$	10.0	99.6	100.0	99.9	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_7$	4.6	99.0	99.3	99.1	99.4	99.2	99.2	99.5	99.3
$\mathcal{U}_8$	0.3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\mathcal{U}_9$	0.1	98.1	99.2	98.5	99.4	99.2	99.1	99.7	99.3

Cuadro 5.9: *TNR* del criterio smh_a.

Podemos concluir que todos los criterios son adecuados en el sentido que seleccionan casi el 100 % de los pares similares. Se comportan bien como propuestas para

resolver de forma aproximada el problema de encontrar los pares similares. Ahora estudiaremos si estos criterios resultan en una ganancia práctica, es decir, si hay un ahorro significativo en el tiempo de búsqueda de pares similares. Para esto se grafican, por cada conjunto de datos, las aceleraciones obtenidas por cada criterios versus la cantidad adicional de memoria utilizada a (variando el tamaño m de los sketches auxiliares).

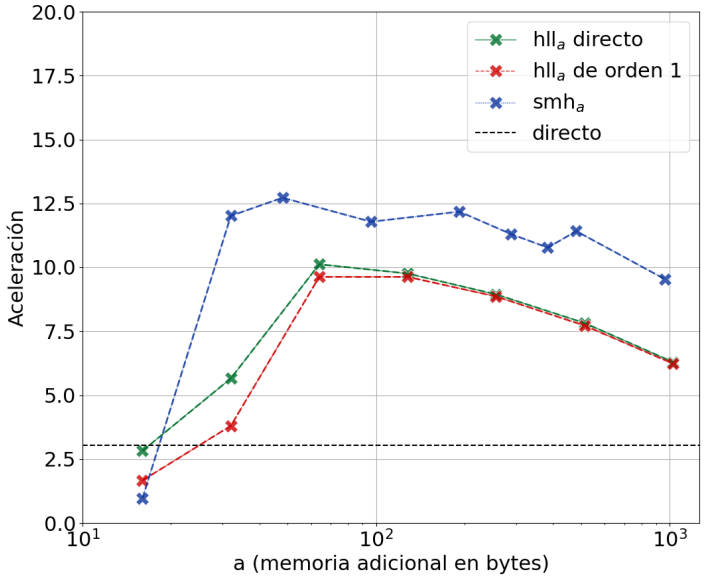


Figura 5.1: Aceleración para el conjunto *archaea*.

Las gráficas se presentan en las figuras 5.1-5.9, en ellas se observa que la aceleración se comporta de manera similar en los distintos conjuntos, sólo que a distinta escala. Los mejores resultados en cuanto a aceleración se consiguen con a cercano a 60 bytes, lo que corresponde a menos del 1 % de memoria adicional con respecto a los sketches principales de 2^{14} bytes. Además, el criterio con el que más tiempo se ahorra es smh_a. En general los criterios que utilizan el sketch de Hyperloglog auxiliar muestran resultados bastante similares. Aunque el criterio hll_a directo suele ser más rápido que el criterio hll_a de orden 1, esto se debe al mayor *TNR* que alcanza este criterio, si bien al costo de tener un menor *TPR* (se pierde una pequeña cantidad

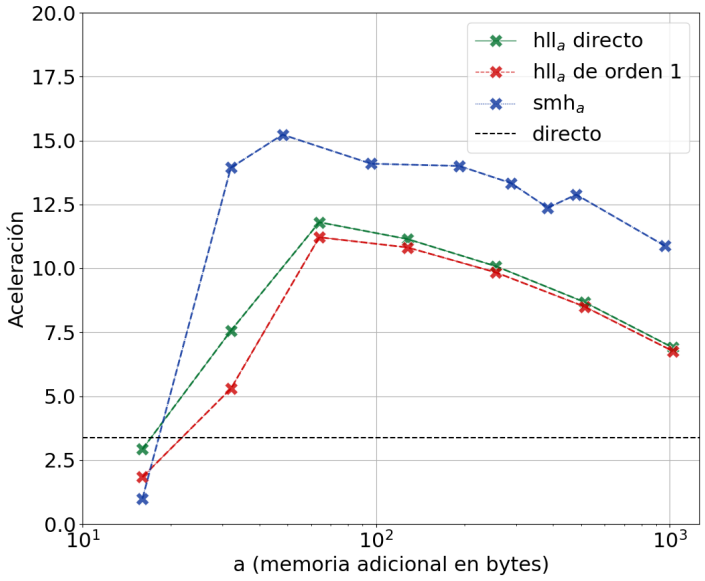


Figura 5.2: Aceleración para el conjunto *fungi*.

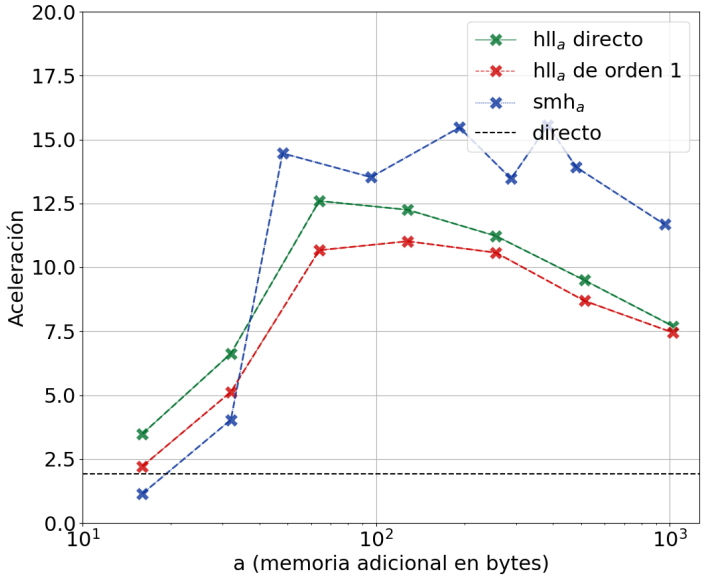


Figura 5.3: Aceleración para el conjunto *vertebrate mammalian*.

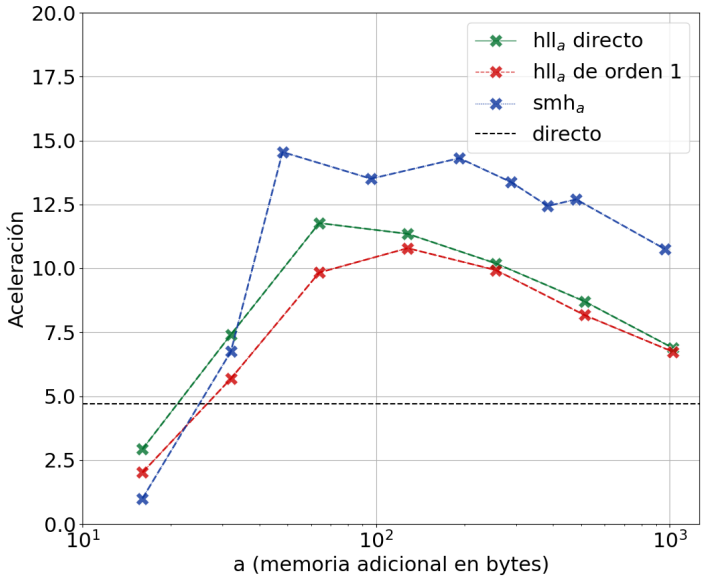


Figura 5.4: Aceleración para el conjunto *vertebrate other*.

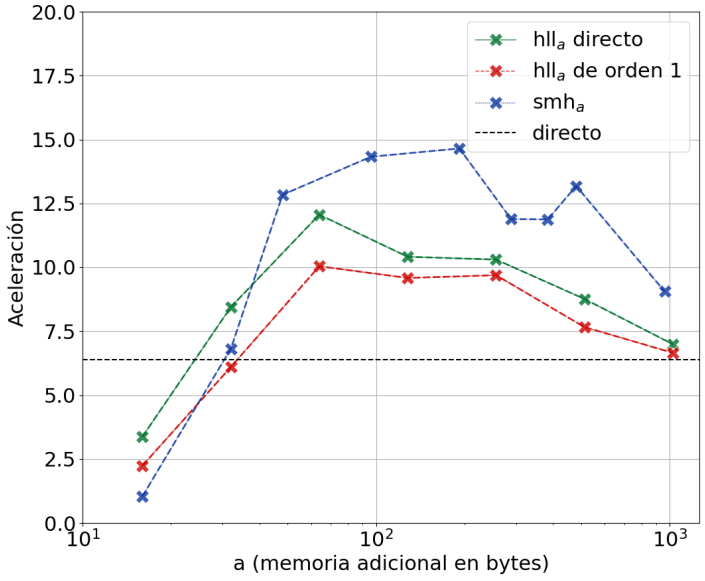


Figura 5.5: Aceleración para el conjunto *plant*.

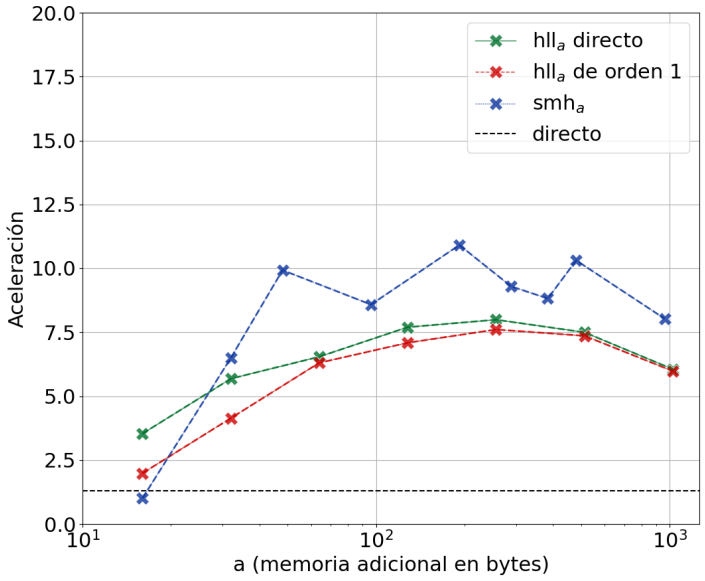


Figura 5.6: Aceleración para el conjunto *prefs*.

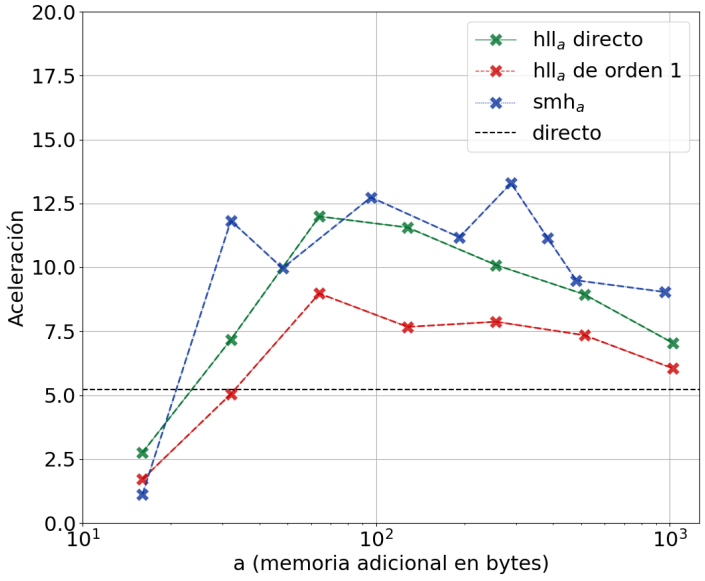


Figura 5.7: Aceleración para el conjunto *protozoa*.

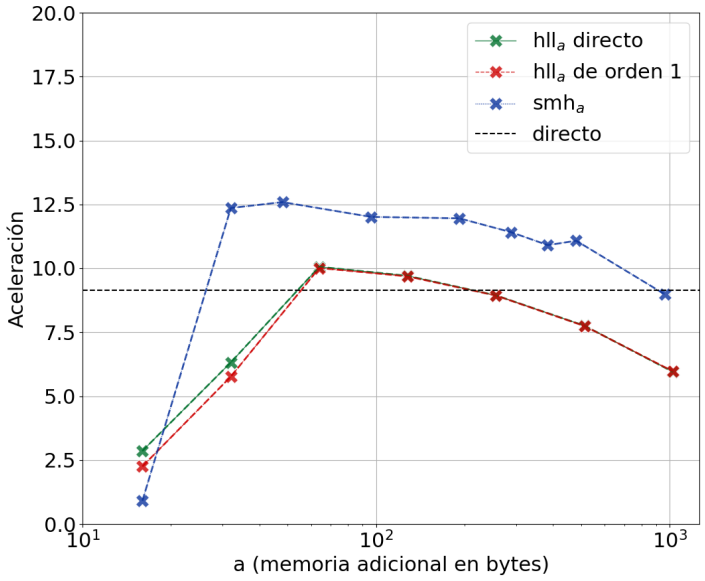


Figura 5.8: Aceleración para el conjunto *viral*.

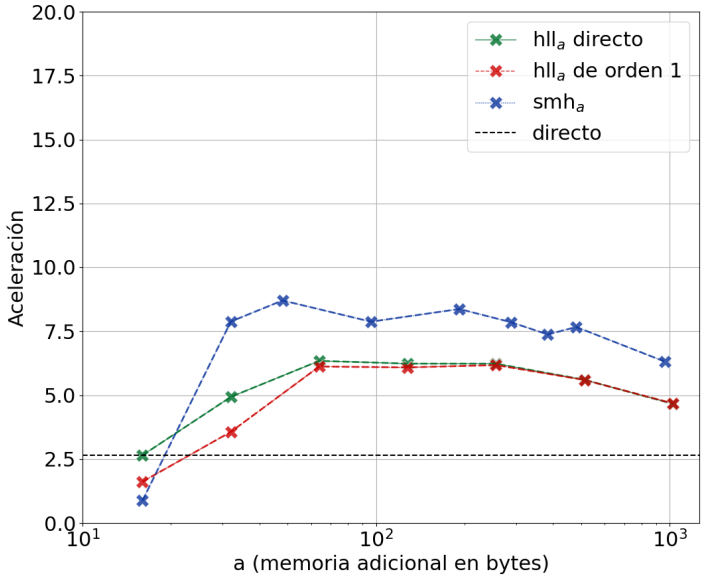


Figura 5.9: Aceleración para el conjunto *bacteria*.

de pares positivos). El criterio directo si bien no utiliza memoria adicional, podemos ver que en general no se ahorra mucho tiempo en comparación a los demás criterios, además de mostrar un comportamiento no tan consistente.

5.5.2. Desempeño para diferentes tamaños de conjuntos de datos

Ahora se estudiará qué ocurre con los criterios cuando el tamaño de los conjuntos de datos es distinto, es decir, se mantiene el umbral constante de $h = 0.8$ y se fija el uso de memoria adicional para los criterios, los que varía ahora es el tamaño del conjunto de datos a procesar. Se utilizó el conjunto de *bacteria*, ya que es el conjunto con mayor cantidad de secuencias y esto permite seleccionar subconjuntos de distintos tamaños. En este caso se construyeron conjuntos de tamaño 10 mil, 20 mil, y así hasta 100 mil, para ello se muestreó sin reposición sobre la lista de secuencias de *bacteria*.

Igual que en la sub-sección anterior, primero se estudia cómo se comporta el *TPR* y *TNR*. Los resultados se entregan en la figura 5.10. Si bien el *TPR* pareciera variar bastante a medida que cambia el tamaño del conjunto, la realidad es que la escala no comienza desde el 0, sino que desde el 0.999. Si se mostraran en un rango mayor (como en el *TNR*), se observarían como líneas horizontales prácticamente iguales a 1. El *TNR* igual se comporta de forma casi constante para los distintos tamaños del conjunto de datos, sólo el criterio directo exhibe un valor bajo cercano a 0.7. Dado que los distintos conjuntos se obtuvieron de muestrear del conjunto de datos *bacteria*, era de esperarse que las métricas de clasificación no variaran realmente.

Por otro lado, la aceleración de los criterios pareciera sí verse afectada por el tamaño del conjunto de datos. En la figura 5.11 se muestran dos gráficos, el primero muestra el tiempo total de encontrar los pares similares de cada criterios (en segundos) en escala log-log, y el segundo, la aceleración lograda por cada criterio. Recordemos que encontrar los pares similares es de orden cuadrático, y esto se refleja en las distintas curvas del primer gráfico, las cuales son rectas de pendiente cercana a 2. Sin importar el criterio utilizado, o incluso si no se utiliza ningún criterio, el tiempo total de

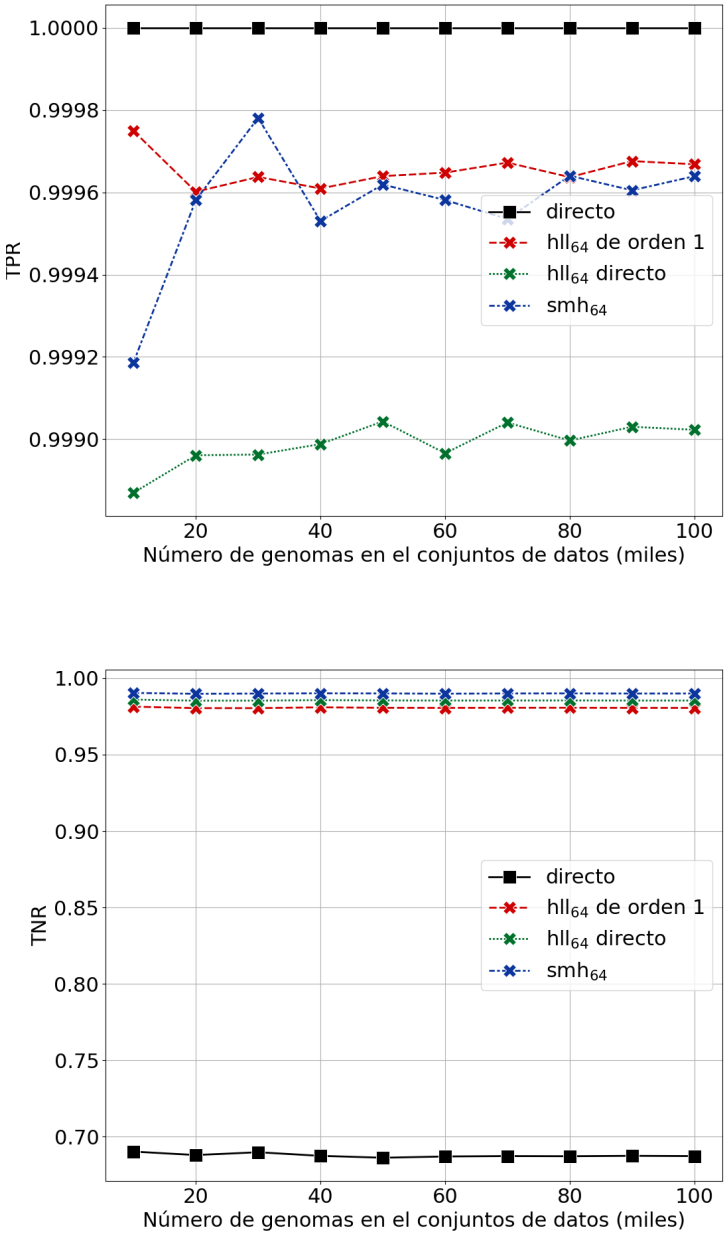


Figura 5.10: *TPR* y *TNR* para conjuntos de datos con diferente cantidad de secuencias.

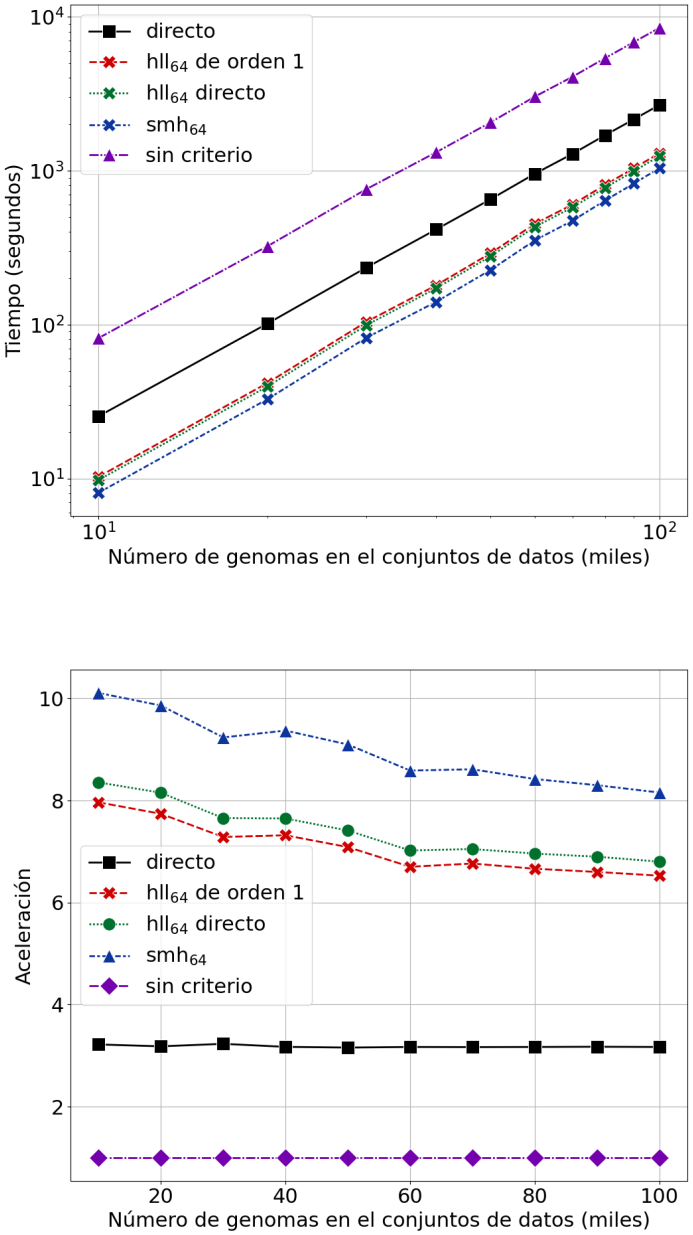


Figura 5.11: Tiempo y aceleración para conjuntos de datos con diferente cantidad de secuencias.

búsqueda crece cuadráticamente con respecto al tamaño del conjunto, sin embargo, utilizar los criterios reducen drásticamente el tiempo, esto se ve en el segundo gráfico. El uso de criterios muestra aceleraciones cercanas a 7 u 8, es decir, se encuentran los pares similares 7 u 8 veces más rápido que si no se utilizara ningún criterio. Si bien pareciera que la aceleración disminuye a medida que crece el conjunto de datos, lo hace cada vez más lentos, dando a entender que pareciera haber un límite de cuánta aceleración se puede alcanzar para conjuntos suficientemente grandes. La ganancia en tiempo sigue siendo considerable.

5.5.3. Desempeño para diferentes valores de h

La elección de $h = 0.8$ es un tanto arbitraria. Se buscaba pedir una alta similitud, sin ser tan exigentes. Ahora se estudiará cómo se comportan los criterios para distintos valores del umbral h, comprobando si es que los criterios se pueden aplicar con distintos umbrales. Para realizar este experimento se consideraron sketches auxiliares de Hyperloglog de tamaño 2^6 y 2^8, y de SuperMinHash de tamaño 8 y 32, buscando los tamaños de sketches de Hyperloglog que mejor resultados daban, y utilizando sketches de SuperMinHash que sean iguales en cuanto a memoria adicional utilizada. Estas elecciones corresponden a $a = 64$ y $a = 254$ bytes de memoria adicional. Así, por cada h se utilizaron 8 criterios: sin criterio, criterio directo, hll_{64} de orden 1, hll_{256} de orden 1, hll_{64} directo, hll_{254} directo, smh_{64} y smh_{256}. Los criterios fueron aplicados al conjunto de datos *prefs* ya que es un conjunto con secuencias genómicas de variados orígenes.

En el primer gráfico de la figura 5.12 se presenta cómo varía el *TPR* para distintos valores de h. Para $h = 0.5$ el criterio smh_{256} baja a un *TPR* bajo 0.92, esto puede ocurrir por la dificultad de encontrar valores de r y b adecuados para este umbral o factores aleatorios dados por la semilla configurada para llevar a cabo los experimentos. Dejando de lado este caso, en todos los demás los criterios se muestra un *TPR* muy cercano a 1, como se esperaría. Por otro lado, en el segundo gráfico de la misma figura se muestra el *TNR*. Todos los criterios parecieran hacer crecer su *TNR* a medida que crece el valor de h. El criterio directo parte bajo y crece muy

lentamente, teniendo un punto de inflexión para $h = 0.5$, donde sube rápidamente, aún así alcanza un TNR de sólo 0.6 para $h = 0.9$. Los demás criterios aumentan su TNR rápidamente inicialmente, alcanzando un TNR superior a 0.8 para $h = 0.5$, más aún, los criterios smh_a alcanzan un TNR muy cercano a 1 ya para $h = 0.3$. Esta ventaja de los criterios basados en SuperMinHash puede deberse a la flexibilidad de ajustar los parámetros r y b según el valor de h.

Respecto a la aceleración, esta se muestra en el gráfico de la figura 5.12. Dado que el TNR aumenta junto con el h, es de esperarse que la aceleración también aumente, ya que es mayor la cantidad de pares negativos que se descarta. Rápidamente los criterios que utilizan más memoria alcanzan una aceleración mayor a 2, creciendo hasta alcanzar aceleraciones cercanas a 8 para $h = 0.9$. La aceleración alcanzada por el criterio directo es considerablemente menor que la alcanzada por el resto de criterios. Para $h = 0.8$ y $h = 0.9$ los criterios que utilizan el sketch de SuperMinHash son los que obtienen la mayor aceleración, sin embargo para los distintos h se observa un comportamiento similares entre estos criterios y los que utilizan un sketch de Hyperloglog de tamaño 2^8 (esto todavía corresponde a menos del 2 % de memoria adicional).

Podemos notar que incluso para distintos valores de h el uso de los criterios de selección propuestos logra una aceleración suficiente para justificar su uso, siempre que se utilice suficiente memoria adicional. Aunque como se mostró, con menos del 2 % de memoria adicional es suficiente. El criterio directo es el único que parece no ser muy confiable en cuanto a la aceleración, sin embargo tiene la ventaja de seleccionar a todos los pares positivos, mas, los nuevos criterios propuestos seleccionan cerca del 99 % de los pares positivos.

5.5.4. Conjunto de datos balanceado

Si bien la experimentación se hizo sobre conjuntos de datos reales, también es interesante ver qué pasa en conjuntos de datos fabricados. Por ejemplo, para los conjuntos estudiados se cumple que la mayoría de los pares tienen un coeficiente de Jaccard menor a 0.5, y los que tienen un coeficiente de Jaccard mayor que 0.9 corresponden

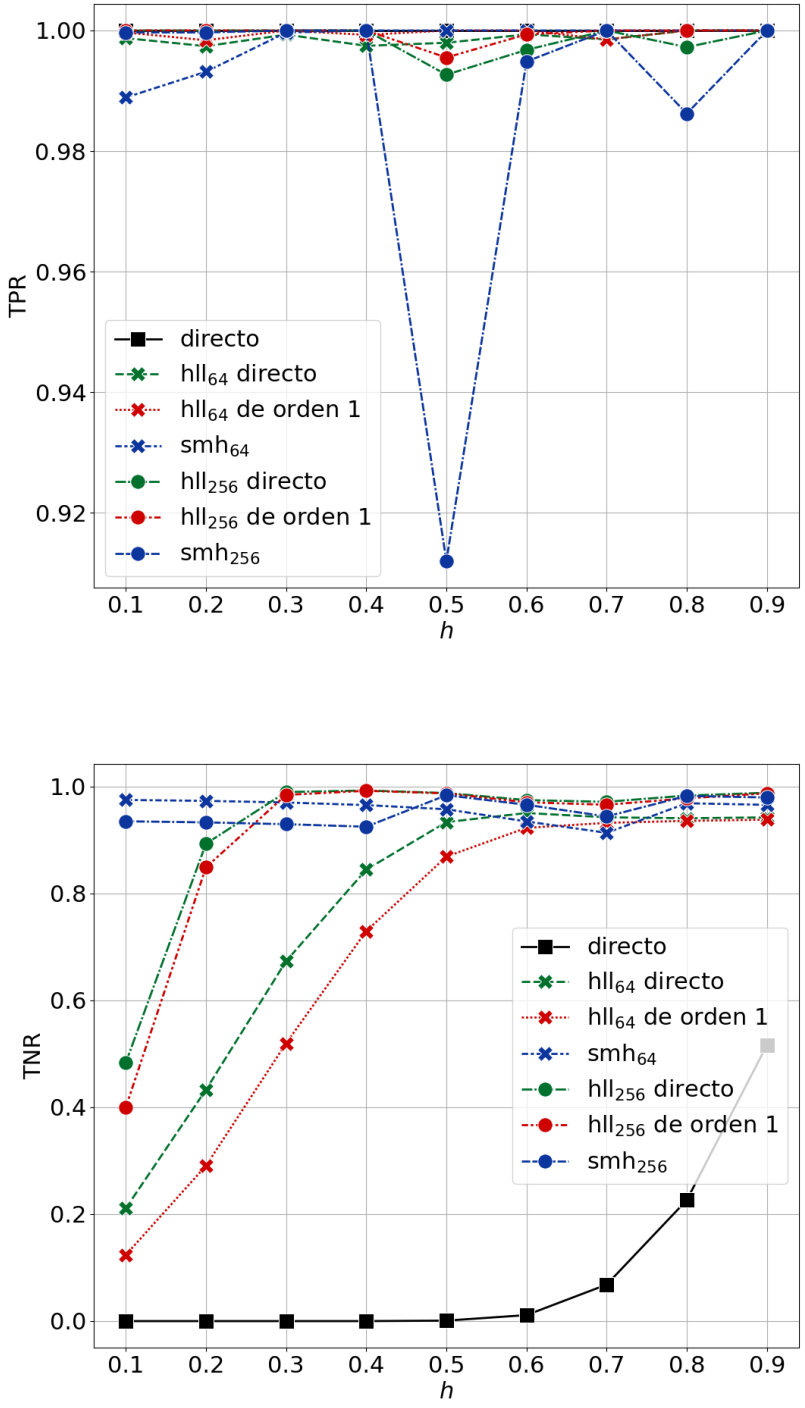


Figura 5.12: *TPR* y *TNR* para diferentes valores del umbral h.

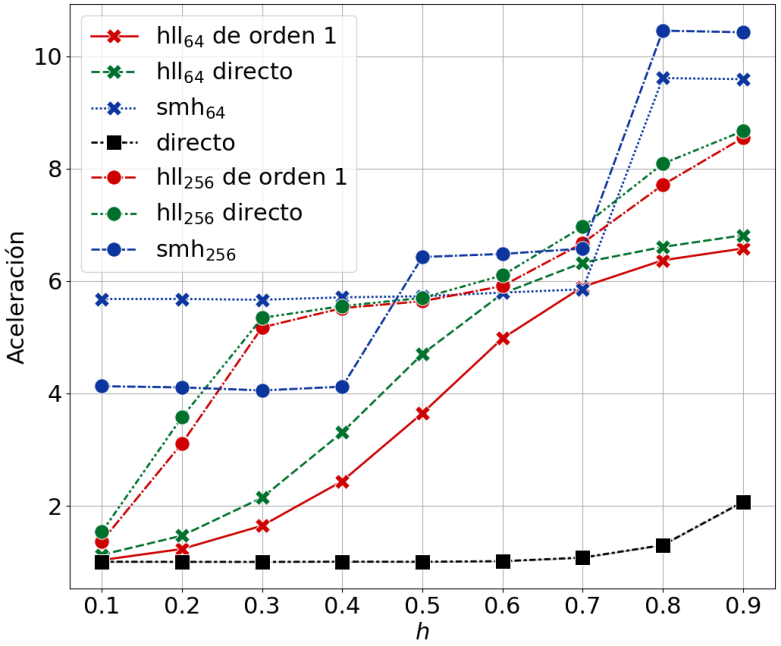


Figura 5.13: Aceleración de los criterios para diferentes valores del umbral h.

a cerca del 1 % de los datos (justamente esta es la razón por la que es relevante descartar los pares negativos). Ahora se pondrán a prueba los distintos criterios con un conjunto de datos que llamaremos *balanced* y denotaremos por U_b, el cual contiene cien mil pares de modo que

$$\forall i \in \{0, 1, 2, .., 9\} \ : \ |(S_1, S_2) \in U_b \ : \ J(S_1, S_2) \in [0.1i, 0.1(i+1)]| = 10000,$$

es decir, hay diez mil pares de secuencias cuyo coeficiente de Jaccard está entre 0 y 0.1, hay diez mil pares de secuencias cuyo coeficiente de Jaccard está entre 0.1 y 0.2, y así hasta los diez mil pares cuyo coeficiente de Jaccard está entre 0.9 y 1. Por esta razón al conjunto se le llamó *balanced*, ya que el valor del coeficiente de Jaccard de los pares se intenta distribuir balanceadamente sobre el rango [0,1].

a	hll_a de orden 1	hll_a directo
16	99.9	99.4
32	99.8	99.5
64	99.9	99.6
128	100.0	100.0
256	99.9	99.8
512	100.0	100.0
1024	100.0	100.0
2048	100.0	100.0
4096	100.0	100.0
8192	100.0	100.0

a	smh_a	directo*
16	100.0	
32	100.0	
48	99.6	
96	99.2	
192	98.7	100.0
288	99.7	
384	100.0	
480	99.0	
960	100.0	
1440	100.0	

Cuadro 5.10: *TPR* de los distintos criterios para *balanced*. * : El criterio directo no utiliza memoria adicional, se agregó a la tabla para que sea más sencilla la comparación con los demás criterios.

Lo primero es estudiar cómo se comportan las métricas de clasificación de los distintos criterios. En el cuadro 5.10 se muestra el *TPR* para los distintos criterios. Similar a los conjuntos estudiados anteriormente, los distintos criterios alcanzan valores de *TPR* cercanos al 100 %, y el criterio directo, como se esperaba, alcanza un 100 % también. Por otro lado, el *TNR*, que se presenta en el cuadro 5.11, sí es bastante distinto a los casos anteriores. Se necesita más memoria para alcanzar altos valores

a	hll_a de orden 1	hll_a directo
16	20.0	30.0
32	30.8	41.3
64	47.4	54.2
128	56.4	62.0
256	73.9	78.7
512	76.6	77.6
1024	78.6	79.1
2048	86.1	86.7
4096	86.3	87.0
8192	93.1	93.3

a	smh_a	directo*
16	0.8	
32	24.7	
48	59.6	
96	46.1	
192	74.2	17.0
288	69.2	
384	59.8	
480	81.6	
960	73.9	
1440	61.1	

Cuadro 5.11: *TNR* de los distintos criterios para *balanced*. * : El criterio directo no utiliza memoria adicional, se agregó a la tabla para que sea más sencilla la comparación con los demás criterios.

del *TNR*. El criterio que muestra peor *TNR* es el criterio directo, con tan sólo 17 %; los demás criterios son capaces de alcanzar valores más altos de *TNR*, pero a costa de más memoria en comparación a los casos anteriores. Pareciera que $a = 256$ es la mejor opción para los criterios que utilizan el sketch auxiliar de Hyperloglog, ya que no es mucha la ganancia de *TNR* con respecto a la memoria adicional para valores mayores de a. En el caso del criterio smh_a, pareciera que desde $a = 192$ se alcanzan valores más altos del *TNR*, pero este no crece junto con a, esto debe ser causado por la dificultad para escojer valores de r y b adecuados, dando saltos entre el porcentaje de filtrado para pares no similares. En este sentido, una ventaja de los criterios basados en Hyperloglog es que la teoría asegura que el criterio mejorará a medida que se aumenta el valor de a, ya que la cota es cada vez más ajustada, a diferencia del criterio smh_a, ya que al usarse el sketch de SuperMinHash sólo para filtrar, la ventaja de utilizar más memoria corresponde a la posibilidad de ajustar mejores valores para los parámetros r y b, pero la mejoría no es tan directa como en el caso de los criterios basados en Hyperloglog, cuyo *TNR* crece conforme a aumenta.

La dificultad en alcanzar valores altos para el *TNR* se ve reflejada en la aceleración, la cual se presenta en la figura 5.14. Además, aunque el *TNR* sea cercano al 100 %,

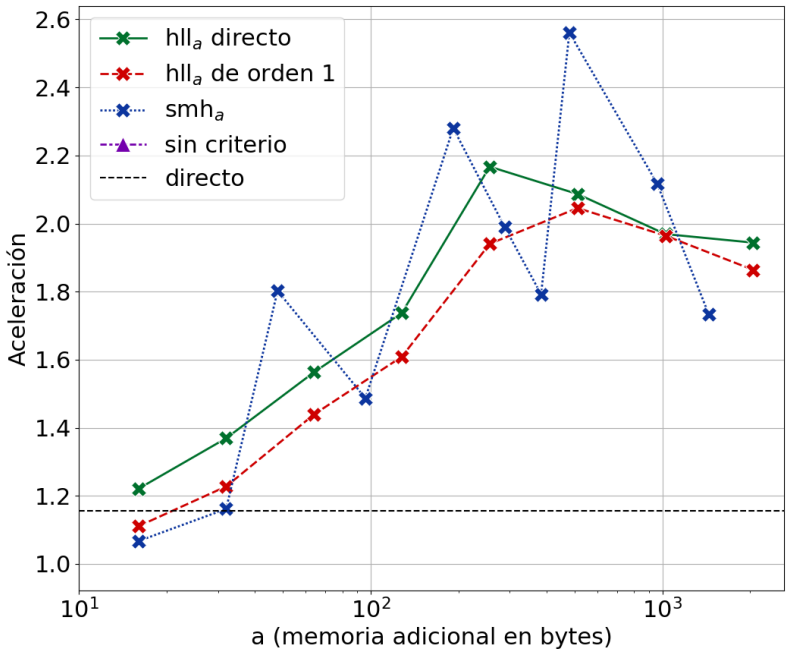


Figura 5.14: Aceleración de los criterios con respecto al uso adicional de memoria de los distintos criterios.

de igual forma la cantidad de pares negativos es mucho menor en comparación a los conjuntos de secuencias $\mathcal{U}_1, ..., \mathcal{U}_9$, por lo que la ganancia en tiempo no es la misma. Aún así, en este caso más balanceado igual se consiguen aceleraciones cercanas a dos, es decir, los criterios propuestos pueden ayudar a resolver el problema en la mitad del tiempo en comparación a no utilizar ningún criterio. De nuevo, el criterio directo se queda atrás con respecto al resto. También se observa que los criterios basados en Hyperloglog se comportan mucho más similar al criterio smh_a, pareciera que la aceleración de este último oscilara al rededor de la aceleración alcanzada por los criterios basados en Hyperloglog.

Similar a como se hizo con los datos del cuadro 5.1, también se experimentó el cómo se comportan los criterios con el conjunto *balanced* cuando se hace variar el umbral h, utilizando los mismos 8 criterios anteriores. Primero, respecto a las métricas de clasificación, estas se muestran en la figura 5.15. El *TPR* sigue siendo cercano al 99 %, excepto para smh_{64} y $h = 0.1$. El bajo valor en este caso puede deberse, de nuevo, a la dificultad para encontrar parámetros r y b adecuados o en la semilla asociada a los experimentos realizados. En cuanto al *TNR*, se observa que los criterios smh_{64} y smh_{256} parten cercanos a 1, pero luego disminuyen a medida que el h alcanza valores cercanos a 0.6, para luego volver a crecer hasta 0.6 para los restantes valores de h. En cambio, los criterios basados en Hyperloglog parten bajos y van aumentando junto con el h, es más, cuando $a = 256$ se observa que incluso superan a los criterios basados en SuperMinHash cuando $h > 0.2$. También las curvas crecen conforme aumenta el a, excepto para smh_a. Se espera entonces que los criterios hll_a (directo y de orden 1) mejoren su *TNR* a medida que aumenta a. El criterio directo parte con un *TNR* muy bajo, y luego se comporta similar a los criterios smh_a, aunque peor. Incluso en un contexto balanceado, el criterio directo no es la mejor opción, y además los criterios basados en Hyperloglog se muestran superiores al utilizar $a = 64$ ($p = 6$) y $a = 256$ ($p = 8$) bytes de memoria adicional por secuencia.

En la figura 5.16 se muestra la aceleración obtenida por cada criterio para distintos valores de h con el conjunto *balanced*. Como se esperaría, la aceleración es bastante baja para valores de h cercanos a 0.1, esto se debe a que en ese caso la mayoría de

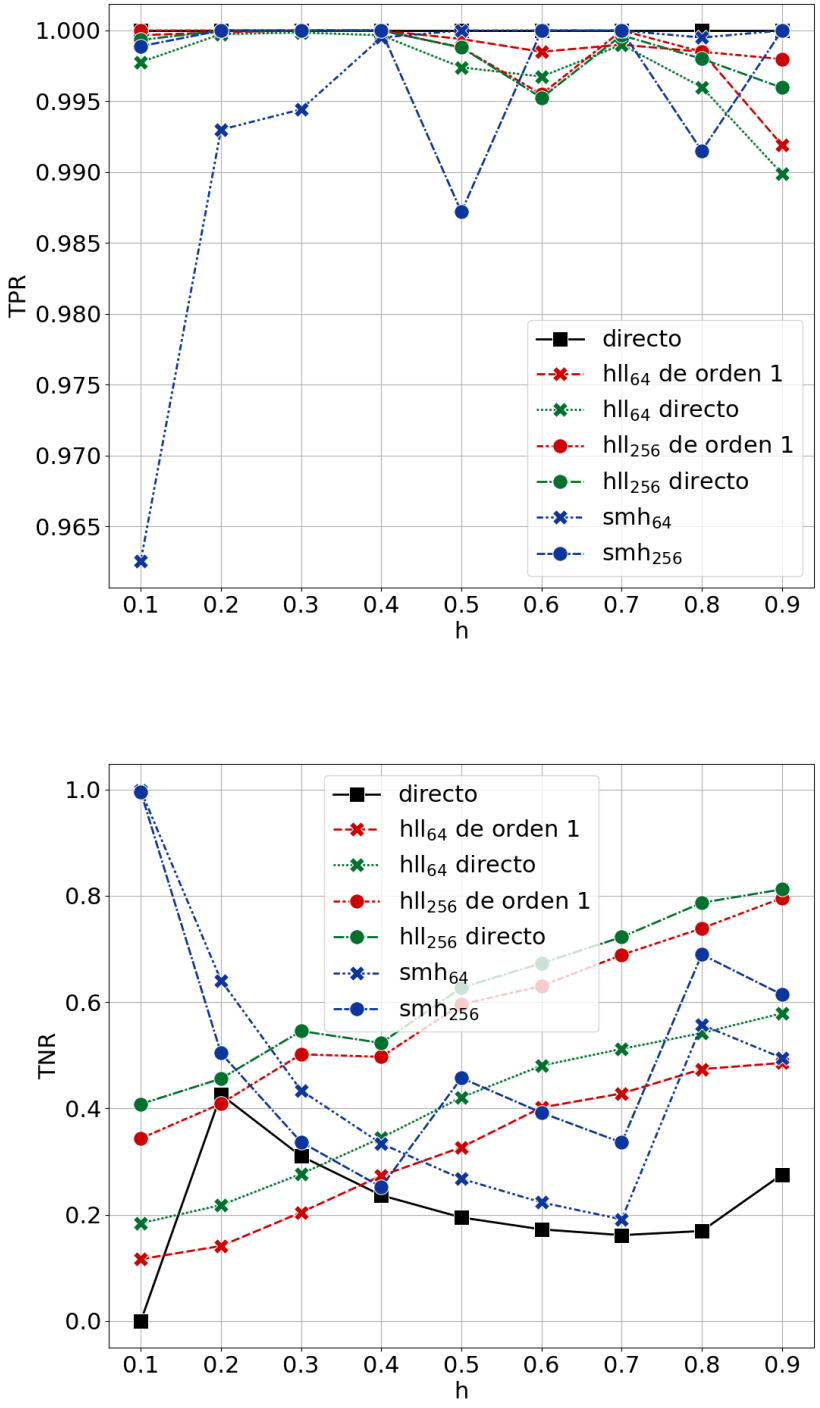


Figura 5.15: *TPR* y *TNR* para diferentes valores del umbral h con el conjunto *balanced*.

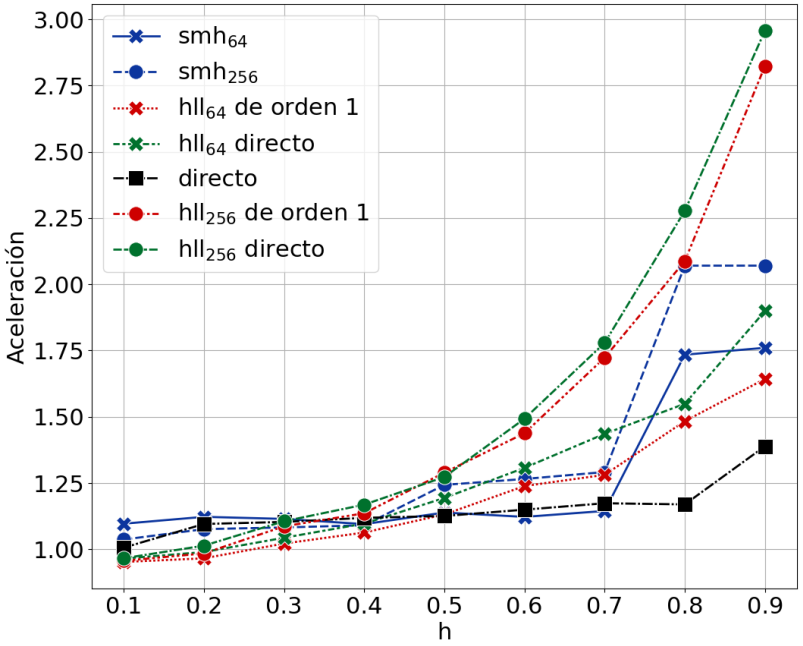


Figura 5.16: Aceleración para diferentes valores de h con el conjunto *balanced*.

los pares son similares, por lo que se ahorra muy poco tiempo al descartar los pares disimilares. Al final se terminan procesando la mayoría de los pares, resultando en que los criterios sean sólo cálculos extras (por esta razón en algunos caso la aceleración es menor a 1). A medida que h crece la porción de pares disimilares también lo hace, provocando que el proceso de descarte efectivamente ahorre tiempo. Todos los criterios aumentan su aceleración a medida que h aumenta, siendo los claros ganadores los criterios basados en Hyperloglog. También se observa que el criterio directo es el que menos aceleración otorga para valores de h suficientemente grandes.

Capítulo 6

Conclusión

6.1. Discusión final

Los tres nuevos criterios propuestos permiten concluir que se cumplió lo hipótesis: fue posible definir nuevos criterios de selección que tengan ventajas considerables sobre el criterio directo, sin presentar grandes desventajas. La principal ventaja es la aceleración, los nuevos criterios son capaces de alcanzar aceleraciones mayores a las del criterio directo. Por otro lado, las únicas dos desventajas son el uso de memoria adicional y el de entregar una aproximación a la solución, sin embargo, experimentalmente se comprobó que ambas desventajas son mínimas: basta con usar cerca del 5 % de memoria adicional para conseguir la mayor aceleración posible para los criterios. Además la sensibilidad es muy cercana a 1, por lo que casi no se pierden pares similares.

Los dos criterios basados en el sketch de Hyperloglog son totalmente nuevos, y el criterios basado en SuperMinHash usa una técnica ya utilizada en este contexto: LSH MinHhash. Si bien para los nueve conjuntos de datos originales $\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_9$ el criterios smh_a suele alcanzar mayores aceleraciones, los criterios basados en Hyperloglog son competitivos. En el esquema de Dashing, en la base de todo se tiene, por cada secuencia genómica, un sketch de Hyperloglog de tamaño lo suficientemente grande

para estimar las cardinalidades con gran precisión. En nuestro caso ese tamaño fue de 2^{14} (cerca de 32KB por sketch). Si ya se tiene este esquema, construir los sketches auxiliares para aplicar los criterios propuestos en este trabajo pareciera exigir volver a procesar cada secuencia para construir los sketches auxiliares, pero esto puede consumir bastante tiempo. Aquí es donde aparece una gran ventaja de los criterios basados en Hyperloglog, y es que a partir del sketch de Hyperloglog principal es posible construir los sketches de Hyperloglog auxiliares, no es necesario volver a procesar la secuencia genómica. Esta es una gran ventaja con respecto a los sketches auxiliares de SuperMinHash, donde es imperante re-procesar cada secuencia. Así, en un entorno donde cada secuencia ya está identificada con su sketch de Hyperloglog principal es posible aplicar los criterios basados en el sketch de Hyperloglog sólo leyendo los sketches principales.

6.2. Trabajo futuro

Tanto el contexto del problema como las soluciones propuestas abren distintas líneas de investigación. La extensión más directa es profundizar en las propiedades de los sketches Hyperloglog y SuperMinHash, además de utilizar nuevos sketches, como SetSketch [7], para definir nuevos criterios de selección.

Por otro lado, el trabajo de esta tesis se contextualizó en la similitud de secuencias genómicas, sin embargo la metodología aplicada puede replicarse sobre palabras de un alfabeto cualquiera, por ejemplo, de páginas web. Rescatar los documentos más similares dentro de una base de datos es una aplicación directa de los criterios de selección.

Además, la similitud de secuencias genómicas se estudió a partir de su representación como conjunto de k-mers y el coeficiente de Jaccard, sin embargo también existe trabajo sobre otras metodologías para valorizar la similitud de secuencias, por ejemplo, a partir de la teoría de la información, o representando a la secuencia a través de vectores en $\mathbb{R}^n$. El mismo coeficiente de Jaccard se puede extender a una versión que considere pesos asociados a cada elemento, ProbMinHash [6] es una clase de

algoritmos que estudian la similitud desde este perspectiva. Así, resulta interesante estudiar cómo se pueden definir los criterios de selección en estos nuevos contextos de similitud genómica.

Por último, uno de los resultados no explorados a profundidad en esta investigación es el de los criterios inversos, aquellos criterios de selección que buscan descartar pares similares y retener todos los pares disimilares. Cuando la mayoría de los elementos en un conjunto de datos son similares y la información relevante está en los pares disimilares, sería posible aplicar estos criterios inversos. Es posible ahondar en su utilidad y rendimiento.

Apéndice

Demostración 6.2.1 ($1-(1-h^{\frac{m}{b}})^b$ es creciente con respecto a b)**.** *Consideremos la función* $f(x) = 1-(1-h^{\frac{m}{x}})^x$*, donde* h, n *son constantes reales. Entonces,*

$$f'(x) = -\underbrace{(1-h^{\frac{m}{x}})^x}_{\geq 0}\left(\underbrace{\ln(1-h^{\frac{m}{x}})}_{<0} + \underbrace{\frac{h^{\frac{m}{x}}}{x(1-h^{\frac{m}{x}})}\ln(h^m)}_{<0}\right),$$

de donde es directo ver que $f'(x) \geq 0$ *para todo* $x \in (0, m]$*. Así, la función* $r \mapsto 1-(1-h^{\frac{m}{b}})^b$ *es creciente con respecto a* b*, con* $b \in (0, m]$*.*

Demostración 6.2.2 (Cota del error de la aproximación de Taylor)**.** *La aproximación* (4.2.4) *tiene un error asociado* $R_n(\hat{t}_p)$ *que está dado por el Teorema de Taylor. Primero notemos que*

$$\begin{aligned} |f_J^{(n+1)}(\hat{t}_p)| &= (n+1)!\frac{(1+\gamma)|A|}{\hat{t}_p^{n+2}} \\ &\leq (n+1)!\frac{(1+\gamma)|A|}{t^{n+2}(1-Z_\alpha\sigma_p)^{n+2}}. \end{aligned}$$

Así, según (2.2.1)*, se deduce que*

$$\begin{aligned} |R_n(\hat{t}_p)| &\leq (n+1)!\frac{(1+\gamma)|A|}{t^{n+2}(1-Z_\alpha\sigma_p)^{n+2}}\ \frac{|\hat{t}_p - t|^{n+1}}{(n+1)!} \\ &\leq \frac{(1+\gamma)|A|t^{n+1}(Z_\alpha\sigma_p)^{n+1}}{t^{n+2}(1-Z_\alpha\sigma_p)^{n+2}} \qquad \text{(Por 4.2.1)} \\ &\leq \frac{2}{(1-Z_\alpha\sigma_p)}\left(\frac{Z_\alpha\sigma_p}{1-Z_\alpha\sigma_p}\right)^{n+1}. \end{aligned}$$

La última desigualdad se explica con que $(1+\gamma) \leq 2$ *y* $|A| \leq t$*. Notar que si* $Z_\alpha \sigma_p \leq 1/2$*, entonces* $\left(\frac{Z_\alpha \sigma_p}{1-Z_\alpha \sigma_p}\right) \leq 1$*, de modo que aumentar el orden disminuiría el error. A modo de ejemplo, si tomamos* $p = 7$ *se tiene que* $|R_1(\hat{t}_p)| \leq 0.1196$*.*

Bibliografía

[1] Daniel N Baker y Ben Langmead. «Dashing: fast and accurate genomic distances with HyperLogLog». En: *Genome biology* 20.1 (2019), págs. 1-12.

[2] Jessica K Bonnie, Omar Y Ahmed y Ben Langmead. «DandD: efficient measurement of sequence growth and similarity». En: *Iscience* 27.3 (2024).

[3] Florian P Breitwieser, Daniel N Baker y Steven L Salzberg. «KrakenUniq: confident and fast metagenomics classification using unique k-mer counts». En: *Genome biology* 19 (2018), págs. 1-10.

[4] Andrei Z Broder. «On the resemblance and containment of documents». En: *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE. 1997, págs. 21-29.

[5] Thomas H Cormen et al. *Introduction to algorithms*. MIT press, 2022.

[6] Otmar Ertl. «Probminhash–a class of locality-sensitive hash algorithms for the (probability) jaccard similarity». En: *IEEE Transactions on Knowledge and Data Engineering* 34.7 (2020), págs. 3491-3506.

[7] Otmar Ertl. «SetSketch: filling the gap between MinHash and HyperLogLog». En: *Proceedings of the VLDB Endowment* 14.11 (2021), págs. 2244-2257.

[8] Otmar Ertl. «Superminhash-A new minwise hashing algorithm for jaccard similarity estimation». En: *arXiv preprint arXiv:1706.05698* (2017).

[9] Philippe Flajolet y G Nigel Martin. «Probabilistic counting algorithms for data base applications». En: *Journal of computer and system sciences* 31.2 (1985), págs. 182-209.

[10] Philippe Flajolet et al. «Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm». En: *Discrete mathematics & theoretical computer science* Proceedings (2007).

[11] Rob Goldbach. «Genome similarities between plant and animal RNA viruses.» En: *Microbiological sciences* 4.7 (1987), págs. 197-202.

[12] Johan Goris et al. «DNA–DNA hybridization values and their relationship to whole-genome sequence similarities». En: *International journal of systematic and evolutionary microbiology* 57.1 (2007), págs. 81-91.

[13] Daniel H Haft et al. «RefSeq and the prokaryotic genome annotation pipeline in the age of metagenomes». En: *Nucleic Acids Research* 52.D1 (2024), págs. D762-D769.

[14] Stefan Heule, Marc Nunkesser y Alexander Hall. «Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm». En: *Proceedings of the 16th International Conference on Extending Database Technology*. 2013, págs. 683-692.

[15] Chirag Jain et al. «High throughput ANI analysis of 90K prokaryotic genomes reveals clear species boundaries». En: *Nature communications* 9.1 (2018), págs. 1-8.

[16] Tomasz Kociumaka, Gonzalo Navarro y Nicola Prezza. «Towards a definitive measure of repetitiveness». En: *Latin American Symposium on Theoretical Informatics*. Springer. 2020, págs. 207-219.

[17] David Koslicki y Hooman Zabeti. «Improving minhash via the containment index with applications to metagenomic analysis». En: *Applied Mathematics and Computation* 354 (2019), págs. 206-215.

[18] Sven Kosub. «A note on the triangle inequality for the Jaccard distance». En: *Pattern Recognition Letters* 120 (2019), págs. 36-38.

[19] Camille Marchet et al. «Data structures based on k-mers for querying large collections of sequencing data sets». En: *Genome Research* 31.1 (2021), págs. 1-12.

[20] Azade Nazi et al. «Efficient estimation of inclusion coefficient using hyperloglog sketches». En: *Proceedings of the VLDB Endowment* 11.10 (2018), págs. 1097-1109.

[21] Brian D Ondov et al. «Mash: fast genome and metagenome distance estimation using MinHash». En: *Genome biology* 17.1 (2016), págs. 1-14.

[22] Anand Rajaraman y Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[23] Raimundo Real y J. Vargas. «The Probabilistic Basis of Jaccard's Index of Similarity». En: *Systematic Biology - SYST BIOL* 45 (sep. de 1996), págs. 380-385. DOI: 10.1093/sysbio/45.3.380.

[24] Javier E Soto, Cecilia Hernández y Miguel Figueroa. «JACC-FPGA: A hardware accelerator for Jaccard similarity estimation using FPGAs in the cloud». En: *Future Generation Computer Systems* 138 (2023), págs. 26-42.

[25] Javier E. Soto et al. «A High-Throughput Hardware Accelerator for Network Entropy Estimation Using Sketches». En: *IEEE Access* 9 (2021), págs. 85823-85838. DOI: 10.1109/ACCESS.2021.3088500.

[26] Zachary D Stephens et al. «Big data: astronomical or genomical?» En: *PLoS biology* 13.7 (2015), e1002195.

[27] Seok-Hwan Yoon et al. «A large-scale evaluation of algorithms to calculate average nucleotide identity». En: *Antonie Van Leeuwenhoek* 110 (2017), págs. 1281-1286.

[28] Andrzej Zielezinski et al. «Alignment-free sequence comparison: benefits, applications, and tools». En: *Genome biology* 18 (2017), págs. 1-17.