Convención Internacional 2017 Universidad Central "Marta Abreu" de Las Villas

Conferencia Internacional de Procesamiento de la Información

Sobre el número máximo de factores frecuentes distintos en una cadena de símbolos
(On the Maximum Number of Distinct Frequent Factors of a String)

M. Baena-García [a] J.M. Carmona-Cejudo [b] J. del Campo-Ávila [c]
G. Ramos-Jiménez [c] R. Morales-Bueno [c]

[a] *Vithas Salud Rincón, 29740 Torre del Mar (Málaga), España*

[b] *ANDATA GmbH, 1050 Viena, Austria*

[c] *Universidad de Málaga, Andalucía Tech, Departamento de Lenguajes y Ciencias de la Computación, 29071 Málaga, España*

Resumen

Las cadenas de símbolos, como fuente de información, siempre han sido un recurso del que poder extraer conocimiento y, actualmente, el número de aplicaciones y casos reales que las usan sigue creciendo, de forma que avances en este ámbito repercutirán en múltiples disciplinas.

En esta comunicación se estudia la complejidad del problema de descubrir factores (subcadenas) frecuentes en cadenas de símbolos de longitud n, añadiendo la característica de que dicha búsqueda pueda estar dirigida por un soporte (frecuencia) k mínimo que deben alcanzar dichos factores. Se analiza cómo afecta este resultado a algoritmos conocidos para este problema y se calcula, de manera efectiva, el número máximo de factores k-frecuentes en una cadena. Se llega a demostrar que, aunque la complejidad en general es cuadrática en la longitud n de la cadena, si el soporte k es al menos $\sqrt{n}$, la complejidad es lineal en n, lo que lo hace suficientemente interesante.

Palabras clave: Minería de cadenas (string mining), descubrimiento de factores (frequent factors), problemas combinatorios (combinatorial problems)

Email addresses: BaenaGM@vithas.es (M. Baena-García), jmcarmona@andata.at (J.M. Carmona-Cejudo), jcampo@uma.es (J. del Campo-Ávila), ramos@lcc.uma.es (G. Ramos-Jiménez), rmorales@uma.es (R. Morales-Bueno).

.

1. Introducción

El estudio y desarrollo de algoritmos de procesamiento de cadenas de símbolos (en adelante cadenas) es una línea de trabajo tradicional en la comunidad científica [1]. Los problemas a resolver han cambiado a lo largo del tiempo debido al aumento de la potencia computacional disponible y debido a nuevos requisitos emergentes de otras áreas científicas tales como extracción de datos biológicos [2]. Por ejemplo, uno de las actividades más destacadas es la secuenciación de ADN que ha generado largas cadenas con información intrínseca aún pendiente de ser descubierta [3].

Las principales tareas que se realizan para estudiar las cadenas y los factores (o subcadenas) que se pueden localizar en ellas [4], se pueden agrupar en dos tipos:

- *indexar* una cadena y *buscar* factores en ella, y
- *descubrir* factores interesantes en cadenas.

En el primer grupo de tareas, el experto plantea una cuestión del tipo: ¿cuán interesante es el factor p en la cadena S? Pero para el segundo tipo de tareas, el tipo de pregunta cambia a: ¿cuáles son los factores interesantes en la cadena S? Esto es, en este segundo caso el factor es desconocido y estamos interesados en descubrirlo. Esta última tarea tiene como dato de entrada solo la cadena S, y el objetivo es encontrar, en la cadena S, factores *interesantes* previamente desconocidos.

Existen algoritmos eficientes para el primer tipo de tareas. Por ejemplo, para el caso de secuencias biológicas, BLAST (Basic Local Alignment Search Tool) [5] es una familia de algoritmos para comparación de secuencias biológicas. Respecto a la segunda tarea han aparecido nuevos algoritmos en los últimos años. En particular es interesante el algoritmo propuesto por Vilo [6], que se ha aplicado a secuencias de proteínas y de ADN.

En la literatura hay trabajos que estudian el número de factores diferentes en una cadena, sin considerar su soporte (número de veces que un factor aparece repetido). Un estudio a nivel abstracto se puede encontrar en el trabajo de Iványi [7]. Por otro lado, Shallit propuso una solución sencilla para calcular el número máximo de factores distintos en cadenas binarias, sin considerar la frecuencia de dichos factores [8]. Esta solución es extendida a alfabetos arbitrarios (también sin considerar factores frecuentes) en Flaxman y otros [9]. Su objetivo era estudiar cuántos factores diferentes puede tener una cadena y mostrar que existe un máximo y es alcanzable.

Apostolico, Bock and Lonardi [10] estudian factores inusuales, que ocurren mucho o muy poco, en un conjunto de cadenas. Ellos están interesados en la estructura combinatoria de tablas estadísticas usadas para detectar factores inusuales. Muestran que, bajo ciertas condiciones, el número de factores inusuales en un conjunto de cadenas se puede acotar y se puede calcular de forma eficiente en tiempo y espacio. Janson, Lonardi and Szpankowski [11] estudian el comportamiento medio del número de factores distintos y definen el *índice de complejidad* basado en esta cantidad. Las cadenas con bajo índice de complejidad contienen un gran número de factores repetidos (al estilo de la complejidad de Kolmogorov [12]).

Un reciente trabajo que puede tener alguna proximidad a los resultados aquí propuestos es [13].

En este trabajo se estudian factores frecuentes según un umbral, pero con la condición de ser aproximados en el sentido de considerar válidos otros factores parecidos en base a considerar la distancia de edición (cambios por permuta, inserción o eliminación). Desarrollan un algoritmo cuya complejidad depende del tamaño de la cadena de entrada y de la distancia de edición y realizan pruebas experimentales con complejidades reducidas.

En el trabajo antes mencionado, Vilo [6] presenta un algoritmo con complejidad en tiempo $O(n^2)$, donde n es la longitud de la cadena. Él observa que, en la práctica, podemos acelerar el algoritmo fijando un valor mayor para el soporte mínimo. Sin embargo, no proporciona una justificación teórica de esta observación.

El objetivo principal de esta comunicación es estudiar la complejidad del problema de encontrar en una cadena los factores frecuentes diferentes que hay en ella con un soporte mínimo.

Para alcanzarlo, en primer lugar calculamos su número máximo y, a continuación, añadimos la restricción de que los factores no deben solaparse con ellos mismos, y estudiamos la complejidad de este problema modificado. Nuestros resultados justifican la observación de Vilo sobre la reducción de la complejidad en la práctica.

El resto del trabajo está organizado como sigue: en la Sección 2, presentamos la notación utilizada y algunos conceptos y resultados básicos. Describimos el algoritmo de Vilo para descubrir factores frecuentes y discutimos su complejidad. También introducimos los grafos de "de Bruijn" y algunos teoremas relacionados que usaremos en nuestras demostraciones. Seguidamente, en la Sección 3 expresamos formalmente la cuestión a resolver y proporcionamos la complejidad computacional de encontrar factores con un soporte mínimo. Proponemos una versión restringida del problema que descarta factores solapados en la Sección 4. La Sección 5 está dedicada al análisis de la complejidad de este último problema. Acabamos con la Sección 6 de conclusiones.

2. Conceptos previos

En esta sección establecemos algunas notaciones y conceptos básicos. Revisamos el algoritmo de Vilo con el fin de descubrir factores frecuentes en una cadena, y definimos los *grafos de "de Bruijn"* [14], así como algunas propiedades que necesitaremos en nuestras demostraciones más delante.

Sea Σ un conjunto finito de caracteres (o símbolos) llamado *alfabeto*. Una *cadena* (o *palabra*) sobre Σ es una secuencia de elementos de Σ. La longitud de una cadena S es el número de sus elementos, y se denota por $|S|$. Los elementos de S se indexan desde 1 hasta $|S|$, siendo S_i el i-ésimo elemento de S. Existe una cadena de longitud 0, llamada la cadena vacía y denotada por ϵ ($|\epsilon| = 0$). Sea $S = S_1 \dots S_n$ una cadena de longitud n; una *subcadena o factor* de S es una cadena $e = S_{i+1} \dots S_{i+|e|}$, donde $i \geq 0$ y $i + |e| \leq n$. Un factor e tiene soporte $k \in \mathbb{N}$ sobre S si e es un factor en al menos k posiciones de S. Decimos que un factor e es frecuente en S si tiene un soporte mínimo $k_0 \in \mathbb{N}$ en la cadena S .

Sea p una cadena de la forma $uvuvu$, donde u es una cadena no vacía. Entonces, en p se puede encontrar el factor uvu en 2 ocasiones reutilizando la u central. Por eso se dice que p es una cadena con *auto-solape*, y el factor uvu es *2-solapado*. Una cadena *sin-solape* es una cadena donde dos ocurrencias distintas de un mismo factor no se solapan. Esto es, una cadena es sin-solape si no tiene factores de la forma $uvuvu$, con u no vacía.

Para analizar el problema de encontrar todos los factores de una cadena S que aparecen en al menos k localizaciones distintas (soporte), estudiamos la solución de Jaak Vilo. Vilo descubrió una solución con complejidad en tiempo proporcional al número de factores frecuentes y alcanzaba la cota $O(n^2)$, que empíricamente podía mejorar hasta una complejidad en tiempo $O(n)$ [4]. En el caso peor, el algoritmo se ejecuta en tiempo cuadrático en $|S|$. Sin embargo, Vilo comprobó que, en la práctica, al usar soportes grandes, el algoritmo es significativamente más rápido, aunque sin dar una justificación analítica. En este trabajo damos una justificación analítica para esa observación en la Sección 5.

Ahora introducimos los *grafos de "de Bruijn"*, una estructura que usaremos para demostrar nuestros resultados. Históricamente, el primer trabajo sobre estos grados se atribuye a Flye-Sainte Marie en 1894 [15]. Más recientemente, Good [16] y de Bruijn [14] han trabajado en este tipo de grafos en 1946.

Un grafo de "de Bruijn" B_j es un grafo orientado definido sobre palabras de longitud j sobre el alfabeto $\Sigma = \{0, 1\}$, y una relación basada en cómo las palabras se solapan. Mas precisamente, B_j es un grafo que tiene 2^j vértices (con etiquetas de la forma $\{0, 1\}^j$), y 2^{j+1} arcos orientados con etiquetas $0, 1$. Cada arco con origen en el vértice etiquetado $a_1a_2 \ldots a_j$, y fin en el vértice etiquetado $b_1b_2 \ldots b_j$ se etiqueta con b_j, sii $a_2 \ldots a_j = b_1 \ldots b_{j-1}$.

Una sucesión de longitud l en el grafo es una secuencia alternante de distintos arcos y vértices (estos últimos pueden no ser distintos), $v_1, e_1, v_2, \ldots, e_l, v_{l+1}$, donde cada vértice v_i está conectado al vértice v_{i+1} mediante el arco e_i. La longitud de la sucesión es el número de arcos. Si $v_1 = v_{l+1}$ entonces es una sucesión cerrada.

Hay una correspondencia biunívoca entre cada sucesión en el grafo con una cadena en el alfabeto. La cadena correspondiente a una sucesión está formada por la etiqueta del primer vértice concatenada con la etiqueta de cada arco en la sucesión. Por tanto, la sucesión $v_1, e_1, v_2, \ldots, e_l, v_{l+1}$ se corresponde con la cadena $v_1e_1e_2 \ldots e_{l-1}e_l$ de longitud $l + j$.

El siguiente Lema 1 describe una propiedad del gafo de "de Bruijn", tomada del trabajo de Shallit [8], pero originalmente atribuida a Yoeli [17].

Lema 1 *Para cada i con $2^j \leq i \leq 2^{j+1}$, el grafo B_j contiene una sucesión cerrada de longitud i que visita cada vértice al menos una vez.*

Para $i = 2^j$, esto es un ciclo Hamiltoniano, y para $i = 2^{j+1}$, es un ciclo Euleriano.

3. Número máximo de factores frecuentes en cadenas

Dada una cadena S de longitud $|S| = n$, estamos interesados en la complejidad del problema de encontrar sus factores frecuentes. El caso peor se produce cuando se alcanza el máximo posible de factores frecuentes. Por tanto, estamos interesados en conocer el máximo número de factores frecuentes que una cadena de longitud n puede tener. Para calcular este número, comenzamos analizando un problema más sencillo: el máximo número de factores distintos en una cadena (con soporte $k = 1$). Después, extenderemos este resultado a un caso más general añadiendo un soporte mínimo para cada factor ($k \geq 1$). En este caso general puede haber auto-solapes en la cadena, lo cual influye de forma significativa en la solución.

Definimos ahora el problema formalmente.

Definición 1 *Sea S una cadena sobre un alfabeto binario, es decir, $S \in \{0,1\}^*$. Definimos $d(S,k)$ como el número total de factores distintos de la cadena S con soporte mínimo k.*

Ejemplo 1 *Por ejemplo, $d(1011011, 2) = 9$, ya que el conjunto de factores con soporte mínimo 2 es*

$$\{\epsilon, 0, 1, 01, 10, 11, 101, 011, 1011\}.$$

Nota: ϵ, la cadena vacía, es un factor con cualquier soporte de cada cadena; esto es $d(S,k) \geq 1 \; \forall S, k$.

Nota: Las cadenas concretas que usamos en los ejemplos las denominamos S seguidas, al mismo nivel, por un número ($S1$, $S2$, $S3$, etc.).

Definición 2 *Sean n y $k \in \mathbb{N}$. Definimos $D(n,k)$ como el máximo número de factores distintos con soporte k que se puede encontrar en cualquier cadena de longitud n:*

$$D(n,k) = \text{máx}\left\{ d(S,k) \;\middle|\; |S| = n \right\} \tag{1}$$

Estamos interesados en estimar los valores de esta función D, para distintos n y k. Comenzamos analizando un caso sencillo: calcular el valor de $D(n,1)$. La solución inicial fue propuesta por Shallit [8]. Comenzando con esa solución nosotros demostraremos la solución general para el valor de $D(n,k)$.

Teorema 1 *Sea $n \in \mathbb{N}$. Entonces*

$$D(n,1) \leq \sum_{0 \leq i \leq n} \text{mín}(2^i, n - i + 1)$$

Demostración 1 *El número de factores diferentes de longitud i será o bien todas las posibles cadenas en el alfabeto binario (esto es, 2^i) o el número de subcadenas de longitud i (es decir $n - i + 1$).*

Teorema 2 *La cota superior del Teorema 1 se alcanza para todo n.*

Demostración 2 *(esbozo) Dada una longitud de cadena n, tomar el único j que cumple* $2^j \leq n - j \leq 2^{j+1}$ *y considerar el grafo de "de Bruijn"* B_j*. Aplicando el Lema 1, existe una cadena cerrada C de longitud* $n - j$ *que pasa por cada vértice en* B_j *y no repite arcos. Y se obtiene*

$$D(n,1) = \sum_{0 \leq i \leq j} 2^i + \sum_{j < i \leq n} n - i + 1 \tag{2}$$

Nota 1 *La ecuación 2 genera la secuencia 1, 2, 4, 6, 9, 13, 17, 22, 28, 35, 43,* 51 . . . *identificada con id A006697 en The On-Line Encyclopedia of Integer Sequences [18]*

El siguiente teorema es la primera aportación original que se presenta en esta comunicación y demostramos una expresión explícita para $D(n,k)$. La idea central es repetir el máximo número de factores distintos, para lo cual usamos el Teorema 2. Sabemos que la cadena asociada a una sucesión cerrada en B_j que no repite arcos, contiene el máximo número de factores. Entonces, repitiendo k ciclos sobre la misma sucesión, nos lleva a una nueva sucesión que repite los mismos arcos solo k veces. Consideremos la cadena S asociada a esta última sucesión. Entonces, esa cadena tiene todos los factores de longitud n/k con soporte mínimo k. Aunque esta no es la cadena de longitud n con máximo número de factores distintos y soporte mínimo k, ella tiene un papel fundamental en la demostración del teorema.

Teorema 3 *Sean* $n, k \in \mathbb{N}$*. Entonces*

$$\begin{aligned} D(n,k) = \max_{x \in \mathbb{N}} \Big\{ & \sum_{0 \leq i \leq j} 2^i + \sum_{j < i \leq j+r+x \cdot k} d - x + \\ & \sum_{j+r+x \cdot k < i \leq j+r+x \cdot k+d-x} d - x + j + r + x \cdot k - i + 1 \Big\} \end{aligned} \tag{3}$$

donde $r \equiv (n-j)$ (**mod** k)*,* $d = \lfloor \frac{n-j}{k} \rfloor$*, y* j *es el único entero que cumple* $2^j \leq (d-x) \leq 2^{j+1}$*.*

Demostración 3 *Sean* n, k *dados y supongamos* $x = 0$*. Entonces,* j *es el único entero que cumple* $2^j \leq d \leq 2^{j+1}$ *siendo* $d = \lfloor \frac{n-j}{k} \rfloor$*. Consideremos el grafo de "De Brujin"* B_j*.*

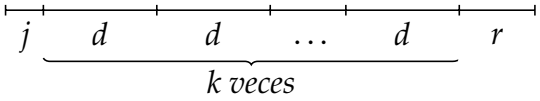
Necesitamos elegir una sucesión cerrada C de longitud d en el grafo B_j*. El lema 1 asegura la existencia de esta sucesión cerrada en C. Esta sucesión no repite arcos, luego todos los factores de la cadena que corresponde a C son distintos.*

Ahora construimos una nueva sucesión C' realizando k iteraciones sobre C y acabando con las primeras $r \equiv (n-j)$ (**mod** k) *parejas arco-vértice en la sucesión C. La longitud de la cadena S que corresponde a la sucesión C' es* $j + k \cdot d + r = n$*, y el factor más largo con soporte k tiene longitud* $l = j + r + d$ *(luego los rangos de los sumatorios son correctos para* $x = 0$*). En este punto, solo necesitamos estudiar el número de factores diferentes en las subcadenas* $S_1 \dots S_l$*. La sucesión C visita cada vértice de* B_j*; luego* $S_1 \dots S_l$ *contienen todos los factores hasta la longitud* j*.*

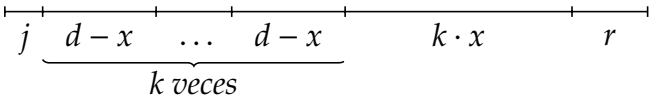
Como repetimos las últimas r parejas arco-vértice en la sucesión correspondiente, obtenemos un número constante "d" de factores de cada longitud desde $j + 1$ *hasta* $j + r$*. Estos factores son auto-solapes en la*

cadena S. Y desde la longitud $j + r + 1$ hasta $j + r + d$, todos los factores son distintos.

En resumen, nuestra cadena es de la forma:



Es posible obtener un valor mayor al considerar $d - x$ en lugar de d. En este caso, la estructura de la cadena deseada S es:



Ahora, al elegir un x apropiado, disminuimos el valor de d en x. También, acabamos las iteraciones con otrasr $k \cdot x$ parejas arco-vértice. Entonces, para longitudes desde $j + r + 1$ hasta $j + r + x \cdot k$, hay un número constante $(d - x)$ de factores diferentes, que representan factores auto-solapados en la cadena S.

Y alcanzamos la expresión de la ecuación 3.

Nota 2 *Los valores iniciales para j y x en un algoritmo para encontrar el máximo valor pueden ser:*

$$\begin{aligned} j_0 &= \lfloor \log_2(n/k) \rfloor \\ x_0 &= \frac{2 \cdot d \cdot k - 2 \cdot r - 2 \cdot d - 1}{4 \cdot k - 2} \end{aligned} \tag{4}$$

Nota 3 *Para $k = 2$, la ecuación 3 genera la secuencia 1, 1, 2, 3, 4, 6, 8, 10, 12, 15, 18, 21, 25, 29, 33, 37, 42, 47, 52, 58, ... (que a fecha de hoy no está incluida en la On-Line Encyclopedia of Integer Sequences)*

4. Factores frecuentes en minería de cadenas

El resultado del Teorema 3 resuelve el primer problema propuesto. Sin embargo, en minería de cadenas el problema debería enunciarse de forma más específica. Cuando analizamos una cadena para descubrir factores, solo deseamos los "suficientemente interesantes". En este sentido, cuando una cadena S tiene un solape $uvuvu$, y el factor auto-solapado uvu es frecuente por el propio auto-solape, entonces ese factor uvu no es interesante. Por este razonamiento, ahora estudiamos las cadenas que maximizan el número de factores frecuentes sin auto-solapes.

Ejemplo 2 *Sea $n = 20$ y $k = 2$. Entonces una cadena $S2$ de longitud 20 con el máximo número de factores frecuentes con soporte mínimo 2 es: 00 101100 101100 101100. En particular,* 1011001 *es uno de los factores que es frecuente por auto-sopale.*

A continuación definimos formalmente el número de factores frecuentes sin auto-solape:

Definición 3 *Sea S una cadena sobre un alfabeto binario y $k \in \mathbb{N}$, $k \leq 2$. Definimos $d_{of}(S,k)$ como el número total de factores distintos de la cadena S que aparecen en k posiciones sin auto-solape.*

Ejemplo 3 *$d_{of}(1011011, 2) = 8$, y el conjunto de factores es $\{\epsilon, 0, 1, 01, 10, 11, 101, 011\}$. El factor 1011 sin auto-solape aparece solo una vez.*

Definición 4 *Sean $n, k \in \mathbb{N}$. Definimos $D_{of}(n,k)$ como el número máximo de factores distintos que aparecen en k posiciones en cualquier cadena de longitud n sin auto-solape:*

$$D_{of}(n,k) = max\Big\{ d_{of}(S,k) \;\Big|\; |S| = n \Big\} \tag{5}$$

Ejemplo 4 *Sea $n = 20$ y $k = 2$. Un ejemplo de una cadena $S4$ de longitud 20 con el máximo número de factores frecuentes con frecuencia al menos 2 sin auto-solape es: 000 101 110 001 011 10001.*

Para esta cadena, $d_{of}(S4, 2) = 49$ que es mayor que $d_{of}(S2, 2) = 34$.

Seguidamente presentamos otro de los resultados de esta comunicación.

Teorema 4 *El número máximo de factores frecuentes no solapados es*

$$D_{of}(n,k) = \max_{z \in \mathbb{N}} \Big\{ \sum_{0 \leq i \leq j} 2^i + \sum_{j < i \leq d-z} d - z + j - i + 1 + \sum_{j+1 \leq i < min\{d-z,\, j+r+z\cdot k+1\}} d - z - i \Big\} \tag{6}$$

donde $r \equiv (n-j)$ $(\boldsymbol{mod}\ k)$, $d = \lfloor \frac{n-j}{k} \rfloor$, y j es el único entero que cumple $2^j \leq (d-z) \leq 2^{j+1}$.

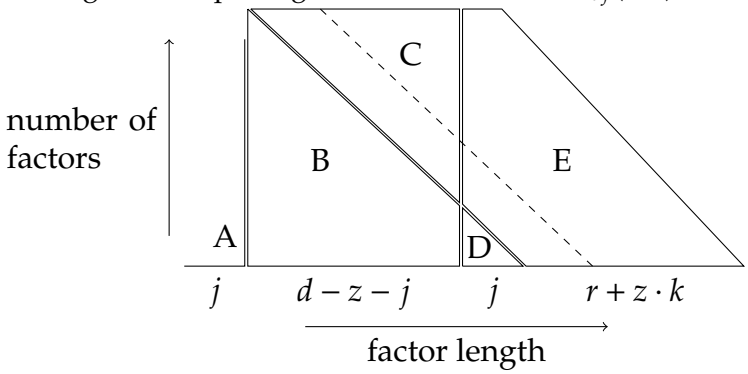
Demostración 4 *Usamos la Figura 1 para seguir el razonamiento. Esta figura representa el aspecto general de la función $D_{of}(n,k)$. Necesitamos contar los factores en las zonas A, B y C (las zonas D y E representan factores frecuentes por auto-solape).*

Como en el Teorema 3, obtenemos la cadena al iterar sobre una sucesión cerrada que visita todos los vértices del grafo B_j. Ahora la función objetivo es distinta. En el Teorema 3 contamos los factores en todas las zonas. Ahora debemos contar solo en las zonas A, B y C. Tenemos un problema similar de maximización, pero con una función objetivo distinta.

En la Ecuación 6 cada término representa una de las zonas A, B o C.

Nota 4 *La Ecuación 6 genera, para $k = 2$, la secuencia 1, 1, 2, 2, 4, 5, 5, 8, 9, 12, 14, 15, 19, 21, 25, 28, 32, 36, 40, 45, 49, 54, 59, 64, 70, 75, 82, 88, 95, 102, . . . (que a fecha de hoy no está incluida en la On-Line Encyclopedia of Integer Sequences)*

Figura 1. Aspecto general de la función $D_{of}(n,k)$



Para acabar esta sección, observamos que los resultados de las secciones 3 y 4 se pueden generalizar a cadenas sobre alfabetos de m símbolos. En este caso las expresiones para $D^m(n,k)$ y $D^m_{of}(n,k)$ son:

$$D^m(n,k) = \max_{x\in\mathbb{N}} \Big\{ \sum_{0\le i\le j} m^i + \sum_{j<i\le j+r+x\cdot k} d-x+ \sum_{j+r+x\cdot k<i\le j+r+x\cdot k+d-x} d-x+j+r+x\cdot k-i+1 \Big\} \tag{7}$$

donde $r \equiv (n-j)$ (**mod** k), $d = \lfloor \frac{n-j}{k} \rfloor$, y j es el único entero que cumple $m^j \le d-x \le m^{j+1}$.

$$D^m_{of}(n,k) = \max_{z\in\mathbb{N}} \Big\{ \sum_{0\le i\le j} m^i + \sum_{j<i\le d-z} d-z+j-i+1+ \sum_{j+1\le i<min\{d-z,j+r+z\cdot k+1\}} d-z-i \Big\} \tag{8}$$

donde $r \equiv (n-j)$ (**mod** k), $d = \lfloor \frac{n-j}{k} \rfloor$, y j es el único entero que cumple $m^j \le d-x \le m^{j+1}$.

5. Complejidad

En esta sección demostramos los dos resultados centrales de este trabajo: el primero analiza la complejidad del problema de la minería de factores en cadenas y su demostración hace uso de los resultados de la sección anterior; el segundo resultado es consecuencia del primero y demuestra formalmente la conjetura de Vilo.

Teorema 5 *El problema de descubrir factores frecuentes en cadenas (sobre alfabetos de m elementos) de longitud n que aparecen en k posiciones sin solape tiene complejidad* $O(n^2/k^2)$.

Demostración 5 *Alcanzamos este resultado analizando la expresión de $D^m_{of}(n,k)$ obtenida en la Ecuación 8, que se puede expresar como:*

$$D^m_{of}(n,k) = \max_{z\in\mathbb{N}}\Bigg\{ \frac{m^{j+1}-1}{m-1} + \\ min\Bigg((d-z-j)\cdot(d-z), \\ (d-z)\cdot(r+z\cdot k) + \frac{(d-z+j+r+z\cdot k+1)\cdot(d-z-j-r-z\cdot k)}{2}\Bigg)\Bigg\} \tag{9}$$

El valor de j es aproximadamente $\log_m(n/k)$, entonces el primer sumando tiene complejidad $O(n/k)$. Respecto al segundo sumando, si observamos la primera componente del mínimo resulta que es de orden $d \cdot d$ que tiene complejidad $O(n^2/k^2)$.

Sabemos que el algoritmo de Vilo tiene complejidad en tiempo proporcional número de factores frecuentes que aparecen. Por tanto, la complejidad es como máximo $O(n^2/k^2)$. Por otra parte, podemos detectar auto-solapes en tiempo $O(n \log n)$ mediante el método definido por [19]. La composición de ambos algoritmos evita los auto-solapes y la complejidad es la suma de ambas. Por tanto, la complejidad en tiempo de $D^m_{of}(n,k)$ es $O(n^2/k^2)$.

Nota 5 *La función $D^m(n,k)$ resuelve un problema de mayor complejidad que la función $D^m_{of}(n,k)$. Las complejidades respectivas son $O(n^2/k)$ y $O(n^2/k^2)$.*

Teorema 6 *Para valores de soporte k suficientemente grandes el algoritmo que encuentra los factores frecuentes es lineal y el algoritmo que encuentra los que además son sin solape es cuasi-lineal en el tamaño de la cadena de entrada.*

Demostración 6 *Dada una cadena S de longitud n, si consideramos soporte k mayor o igual que $\sqrt{n}$ tenemos que la complejidad del algoritmo que detecta los factores frecuentes pasa a ser $O(n)$ (lineal), y como los auto-solapes se detectan en $O(n \log n)$, resulta que el algoritmo compuesto final tiene complejidad en tiempo $O(n \log n)$ (cuasi-lineal).*

6. Conclusiones

Campos de investigación como la bioinformática generan cadenas muy largas, lo cual hace que sea de interés resolver problemas de *minería de cadenas*. Esto lleva al diseño de nuevos algoritmos que extraigan conocimiento en forma de factores interesantes, pudiéndoles requerir, además, que presenten un soporte mínimo.

En esta comunicación hemos presentado una solución explícita al problema de contar el número

de factores frecuentes, que nos lleva a la complejidad del algoritmo que los encuentra. Hemos dado expresiones de la cota superior y hemos demostrado que es alcanzable. Hemos añadido la condición de evitar el auto-solape; hemos obtenido la expresión para este nuevo problema y hemos demostrado que la complejidad de este problema es $O(n^2/k^2)$. Esto significa que si la frecuencia de interés es $k \geq \sqrt{n}$, entonces el problema es resoluble en tiempo lineal.

Nuestro resultado proporciona una justificación formal de que el algoritmo de Vilo, aunque es cuadrática en teoría, tiene en la práctica complejidad lineal (al considerar soporte alto).

Hemos presentado dos nuevas sucesiones de números naturales, que surgen de considerar distintas longitudes de cadenas y soportes. Estas dos nuevas sucesiones aún no han sido registradas en la "On-Line Encyclopedia of Integer Sequences".

Una implementación de las funciones $D^m(n,k)$ y $D^m_{of}(n,k)$ para contar el máximo número de factores frecuentes está en `http://mbaena-research.appspot.com/frequentfactors`.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el I Plan Propio de Investigación y Transferencia de la Universidad de Málaga.

Referencias

[1] D. Gusfield, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, Cambridge University Press, 1997.

[2] P. Kalsi, H. Peltola, J. Tarhio, Comparison of exact string matching algorithms for biological sequences, in: S. B. Heidelberg (Ed.), Bioinformatics Research and Development, Vol. 13, Springer, 2008, pp. 417–426.

[3] W. Gibbs, The unseen genome: gems among the junk, Scientific American 285 (5) (2003) 46–53.

[4] J. Vilo, Discovering frequent patterns from strings, Tech. rep., Department of Computer Science, University of Helsinki, Finland (1998).

[5] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman, Basic local alignment search tool., J Mol Biol 215 (3) (1990) 403–410. doi:http://dx.doi.org/10.1006/jmbi.1990.9999.

[6] J. Vilo, Pattern discovery from biosequences, Ph.D. thesis, Department of Computer Science, University of Helsinki, Finland (2002).

[7] A. Iványi, On the d-complexity of words, Ann. Univ. Sci. Budapest. Sect. Comput. 8 (1987) 69–90.

[8] J. Shallit, On the maximum number of distinct factors in a binary string, Graphs and Combinatorics 9 (1993) 197–200.

[9] A. Flaxman, A. W. Harrow, G. B. Sorkin, Strings with maximally many distinct subsequences and substrings, Electron. J. Combin. 11 (1).

[10] A. Apostolico, M. E. Bock, S. Lonardi, Monotony of surprise and large-scale quest for unusual words., Journal of Computational Biology 10 (3/4) (2003) 283–311.
URL http://dblp.uni-trier.de/db/journals/jcb/jcb10.html#ApostolicoBL03

[11] S. Janson, S. Lonardi, W. Szpankowski, On average sequence complexity, Theor. Comput. Sci. 326 (1-3) (2004) 213–227. doi:10.1016/j.tcs.2004.06.023.
URL http://dx.doi.org/10.1016/j.tcs.2004.06.023

[12] M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, Springer-Verlag New York, 2008.

[13] J. Shang, J. Peng, J. Han, MACFP: maximal approximate consecutive frequent pattern mining under edit distance, in: Proceedings of the SIAM International Conference on Data Mining, 2016, pp. 558–566.

[14] N. G. de Bruijn, A combinatorial problem, Nederl. Akad. Wetensch. Proc. 49 (1946) 758–764.

[15] C. F.-S. Marie, Solution to problem number 58, L'Intermédiaire des Mathématiciens 1 (1894) 107–110.

[16] I. J. Good, Normally recurring decimals, J. London Math. Soc. 21 (1946) 167–169.

[17] M. Yoeli, Binary ring sequences, Amer. Math. Monthly 69 (1962) 852–855.

[18] N. J. A. Sloane, The on-line encyclopedia of integer sequences, published electronically at www.research.att.com/~njas/sequences/ (2009).

[19] J. Stoye, D. Gusfield, Simple and flexible detection of contiguous repeats using a suffix tree, Theor. Comput. Sci. 270 (1-2) (2002) 843–850. doi:10.1016/S0304-3975(01)00121-9.
URL http://dx.doi.org/10.1016/S0304-3975(01)00121-9