Journal homepage http://revistas.unitru.edu.pe/index.php/SSMM

SELECCIONES MATEMÁTICAS
Universidad Nacional de Trujillo
ISSN: 2411-1783 (Online)
2019; Vol.06(2): 275 - 282.

Una aproximación al problema de la Subsecuencia Común más Larga de Múltiples Secuencias usando entropía de Shannon.

An approximation to the Longest Common Subsequence for Multiple Sequences problem using Shannon's Entropy.

Ross Mary Sáenz *

Received, Jul. 30, 2019 *Accepted, Nov. 24, 2019*

How to cite this article:
Saenz, R. M. *Una aproximación al problema de la Subsecuencia Común más Larga de Múltiples Secuencias usando entropía de Shannon*. Selecciones Matemáticas. 2019; 6(2):275-282. http://dx.doi.org/10.17268/sel.mat.2019.02.13

Resumen

El problema de la Subsecuencia Común más Larga de Múltiples Secuencias (SCLM), ha sido ampliamente estudiado en Ciencias de la Computación por más de 40 años, motivado principalmente por sus diversas aplicaciones en Bioinformática. En este artículo se presenta un algoritmo heurístico que estima una o más soluciones al problema SCLM, utilizando entropía de Shannon como una medida de la información para determinar alineamientos con el mayor numero de coincidencias.

Palabras clave. Algoritmo Heurístico, Subsecuencia, Subsecuencia Común, Alineamiento, Entropía.

Abstract

The Longest Common Subsequence for Multiples Sequences problem (LCSM), has been studied for more than 40 years, mainly motivated by its multiple applications in Bioinformatics. In this article we present a heuristic algorithm that estimates one or more solutions to the LCSM problem, using the Shannon's Entropy as a measure of information in order to determine alignments with the greatest number of matches.

Keywords. Heuristic Algorithm, Subsequence, Common Subsequence, Alignment, Entropy, Simulation.

1. Introducción. El problema de encontrar la *subsecuencia común más larga* de una colección de secuencias dadas, consiste en calcular una subsecuencia de longitud máxima, común a un conjunto de dos o más secuencias. Este problema es actualmente de gran interés en Ciencias de la Computación por sus diversas aplicaciones, especialmente en el área de Biología Computacional.

Por más de 40 años se han propuesto diversas alternativas de solución, en el caso particular de colecciones con dos secuencias, las técnicas más implementadas son, Autómatas Finitos y Punto Dominante [**?**, 3, 5]. Para colecciones de múltiples secuencias, estudios recientes evidencian estrategias conjuntas tales como, Punto Dominante y Paralelismo, entre otros [6, 7].

En este artículo se describe un algoritmo heurístico que aproxima una solución al problema SCLM y usa una medida de la información conocida como la entropía de Shannon. El principal aporte de este método es estudiar una técnica novedosa que estime una solución al problema, capturando la información común de una colección de secuencias dadas.

Es preciso señalar que este problema es de complejidad NP-Hard [1, 8], por tal razón es razonable proponer un algoritmo heurístico que permita calcular una solución aproximada al problema.

*Facultad de Ciencias, Universidad Nacional de Colombia, Bogotá-Colombia (rmsaenzl@unal.edu.co).

2. Definiciones. Un *alfabeto* Σ es un conjunto finito no vacío, cuyos elementos son símbolos o caracteres, y cuyo cardinal se representa por $|\Sigma|$. Una *secuencia* A sobre Σ se define como un arreglo ordenado de longitud finita de elementos de Σ, la longitud de la secuencia A se denota por $|A|$. Nótese que la cadena vacía o nula es aquella que no contiene símbolos y se representa por ε. Sean $A = a_1a_2...a_n$ y $C = c_1c_2...c_r$ dos secuencias sobre un alfabeto Σ, C es una *subsecuencia* de A, si $r \leq n$ y existe una función monótona creciente $F : [r] \to [n]$, donde $[n] = \{1, 2, \cdots, n\}$ y $[r] = \{1, 2, \cdots, r\}$, tal que $F(i) = k$ si $c_i = a_k$ para $k \in [n]$. En otras palabras, una subsecuencia C es un subconjunto ordenado de caracteres de la secuencia A no necesariamente consecutivos. En general, dada una colección de secuencias $\{s_1, s_2, \cdots, s_n\}$ sobre Σ, se dice que una secuencia C sobre Σ es una *subsecuencia común de* $\{s_1, s_2, \cdots, s_n\}$ si C es subsecuencia de s_r para todo $r \in \{1, 2, \cdots, n\}$. C es una *subsecuencia común más larga*, si C tiene longitud máxima. Por ejemplo, dadas las secuencias $s_1 = CGCGTA$, $s_2 = AGCCGTAC$ y $s_3 = AGTCCGT$, sobre $\Sigma = \{A, C, G, T\}$, las subsecuencias comunes más largas de la colección $\{s_1, s_2, s_3\}$ son $\{CCGT, GCGT\}$.

Un *alineamiento* entre un par de secuencias de caracteres es una correspondencia entre los caracteres de las secuencias dadas, de tal manera que se presente cierta cantidad de coincidencias, por ejemplo una alineación entre $\{abcd, acc\}$ es:

(2.1)
$$\begin{array}{ccccc} a & b & c & d & - \\ a & - & c & - & c \end{array}$$

Los espacios o huecos presentes en el alineamiento se denominan gaps y se representan por el símbolo " $-$ " o " ". Otro alineamiento podría ser:

$$\begin{array}{cccccc} - & - & a & b & c & d \\ a & c & - & - & c & - \end{array}$$

Sin embargo, el mejor alineamiento posible es el dado en (2.1) ya que éste tiene la mayor cantidad de coincidencias. Formalmente, un alineamiento se define de la siguiente forma.

Definición 1. *Sea $A = a_1a_2...a_n$ y $B = b_1b_2...b_r$ secuencias definidas en Σ, donde $r \leq n$. Un alineamiento entre la secuencia A y B es una matriz $M_{2\times m}$ con elementos en el alfabeto $\Sigma \cup \{-\}$, donde $n \leq m \leq r + n$, tal que:*

- *La primera fila de la matriz M contiene los caracteres de A y la segunda los de B, en orden (por conveniencia se toma en la primera fila la secuencia con mayor longitud).*
- *Cada columna de la matriz contiene por lo menos un símbolo del alfabeto Σ , es decir las columnas de la matriz están formadas por parejas de la forma: $\begin{pmatrix} a_i \\ - \end{pmatrix}$, $\begin{pmatrix} - \\ b_j \end{pmatrix}$ y $\begin{pmatrix} a_i \\ b_j \end{pmatrix}$ si $a_i = b_j$, donde $1 \leq i \leq n$ y $1 \leq j \leq r$.*

En general, los gaps permiten una mejor distribución de los caracteres con el fin de que el alineamiento contenga el mayor número de coincidencias.

En términos generales, *un alineamiento múltiple* corresponde al alineamiento simultáneo de tres o más secuencias. Es decir, un alineamiento para $\{s_1, s_2, \cdots, s_n\}$ sobre un alfabeto Σ es una matriz $M_{n\times m}$, de modo que:

- Cada entrada de la matriz M pertenece a $\Sigma \cup \{-\}$.
- m cumple con $max\{|s_1|, |s_2|, \cdots, |s_n|\} \leq m \leq \Sigma_{i=1}^{n}|s_i|$.
- Toda columna de M contiene por lo menos un símbolo de Σ.
- La fila $k-$ésima de la matriz M contiene los caracteres ordenados de una secuencia s_k, separados por g_k gaps, donde $g_k = max\{|s_1|, |s_2|, \cdots, |s_n|\} - |s_k|$, donde $1 \leq k \leq n$.

En conclusión, *un alineamiento múltiple de $\{s_1, s_2, \cdots, s_n\}$ con g gaps*, es un alineamiento de $\{s_1, s_2, \cdots, s_n\}$ con $m = l_{max}$, donde $l_{max} = max\{|s_1|, |s_2|, \cdots, |s_n|\} + g$. Por ejemplo, un alineamiento múltiple entre el conjunto de secuencias $\{abcd, acc, acd, ad\}$ con 1 gap, está dado por:

$$\begin{array}{ccccc} a & b & c & d & - \\ a & - & c & - & c \\ a & - & c & d & - \\ a & - & - & d & - \end{array}$$

Note que 1 gap le corresponde a la secuencia de mayor longitud, es decir a la secuencia $abcd$, 2 gaps para acc y acd, y 3 gaps para ad.

Definición 2. *Una nube de secuencias es una matriz que contiene todas las formas posibles de organizar una secuencia* $s = \{a_1, a_2, \cdots, a_l\}$ *con g gaps en* $l+g$ *lugares, de tal modo que los* $\{a_1, a_2, \cdots, a_l\}$ *caracteres preserven el orden de aparición .*

Observe que el número de formas de organizar los caracteres $a_1, a_2, \ldots, a_l$ en $l+g$ lugares está dado por $\binom{l+g}{g}$.

Por ejemplo, la nube se secuencias de $s = \{ab\}$ con $g = 2$, es la siguiente:

a	b	-	-
a	-	b	-
a	-	-	b
-	-	a	b
a	-	-	b
-	a	b	-

maier1978complexity

Entropía. A continuación se define la función de entropía de Shannon como una medida de la información; ésta mide el grado de desorden o incertidumbre que se genera en los resultados de un experimento cualquiera, también se puede interpretar como la cantidad de información requerida en promedio para describir una variable aleatoria. En este orden de ideas, si se considera el siguiente arreglo de símbolos, la entropía de que, el símbolo **a** aparezca en una posición dada de este arreglo es mayor que la del símbolo **b**.

a	a	
a		a
	a	a
a	b	
	a	b
a		b

Definición 3. *[2] Sea* X *una variable aleatoria con distribución de probabilidad* $P(X)$*, si* X *toma valores en* $\{x_1, x_2, \cdots, x_n\}$*, la entropía* $H(X)$ *está definida por:*

$$\mathrm{H}(X) = -\sum_{i=1}^{n} p(x_i)\, log_2 p(x_i)$$

2.1. Matriz Combinatoria de S**.** Sea Σ el alfabeto de símbolos en S y m el cardinal de Σ; además, sean $l_1, l_2, \ldots, l_n$ las longitudes de $s_1, s_2, \ldots, s_n$ respectivamente, se considera $l_{max} = max\{|s_1|, |s_2|, \cdots, |s_n|\} + g$. Así, para cada s en S, la *Matriz Combinatoria* de s está definida por la matriz de m filas y l_{max} columnas, cuyas filas son indexadas por los símbolos de Σ en orden alfabético, y las columnas por las posiciones 1 hasta l_{max}, de tal manera que la entrada $(\mathbf{x}, j)$ de la matriz, contiene el número de secuencias en la nube de s con $l_{max} - |s|$ gaps que tienen el símbolo **x** en la posición j, donde $1 \leq j \leq l_{max}$. Por último, se dice que la *Matriz Combinatoria* de S es la suma de las matrices combinatorias de cada s en S.

Por ejemplo, sea $S = \{abab, bcca, cab\}$ y $g = 2$, las matrices combinatorias de cada s en S se muestran a continuación.

Para $s_1 = abab$, la nube de secuencias tiene 2 gaps, luego la matriz combinatoria está dada por:

	1	**2**	**3**	**4**	**5**	**6**
a	10	4	4	6	6	0
b	0	6	6	4	4	10
c	0	0	0	0	0	0

La nube de secuencias para $s_2 = bcca$ contiene 2 gaps, luego la matriz combinatoria está dada por:

	1	**2**	**3**	**4**	**5**	**6**
a	0	0	0	1	4	10
b	10	4	1	0	0	0
c	0	6	9	9	6	0

De igual manera, para la secuencia $s_3 = cab$ la nube de secuencias tiene 3 gaps, luego la matriz combinatoria es:

	1	**2**	**3**	**4**	**5**	**6**
a	0	4	6	6	4	0
b	0	0	1	3	6	10
c	10	6	3	1	0	0

La suma de las matrices anteriores corresponde a la matriz combinatoria de $S = \{abab, bcca, cab\}$ con $g = 2$ y está dada por:

	1	**2**	**3**	**4**	**5**	**6**
a	10	8	10	13	14	10
b	10	10	8	7	10	20
c	10	12	12	10	6	0

CUADRO 2.1
Matriz Combinatoria

2.2. Secuencias Patrón. Sea $M = [m_{\mathbf{x}j}]$ la matriz combinatoria de S, para una columna j de la matriz M se define la frecuencia máxima de j como:

$$frecMax(j) := \max_{\mathbf{x}\in\Sigma} m_{\mathbf{x}j}.$$

Es decir, el máximo de los valores en la j-ésima columna.

Una *Secuencia Patrón* de M es una secuencia de longitud l_{max} con símbolos en el alfabeto Σ, donde el j-ésimo caracter de la secuencia corresponde a un símbolo **a**, tal que $frecMax(j) = m_{\mathbf{a}j}$.

Sea $P = \{p_1, p_2, \cdots, p_w\}$ el conjunto de todas las secuencias patrón de M. Para la matriz combinatoria de la Tabla 2.1 , el conjunto de secuencias patrón es

$$P = \{accaab, bccaab, cccaab\}.$$

Observe que para el ejemplo anterior, se obtienen tres secuencias patrón, ya que la frecuencia máxima de la columna uno ocurre en los símbolos $\{a, b, c\}$.

2.3. Entropía de símbolos y posiciones. Dada una matriz combinatoria M de la colección de secuencias S y un símbolo **x** en el alfabeto Σ, *la entropía de un símbolo* ***x*** se define como:

$$\mathrm{H}(\mathbf{x}) := -\sum_{j=1}^{l_{max}} P(\mathbf{x}, j) \cdot \log_2(P(\mathbf{x}, j)).$$

Donde $P(\mathbf{x}, j)$ es la probabilidad de que dada una ocurrencia del símbolo **x** en una secuencia, ésta se encuentre en la posición j. Observe que la probabilidad $P(\mathbf{x}, j)$ está dada por la expresión $P(\mathbf{x}, j) = \frac{m_{\mathbf{x}j}}{\sum_{j=1}^{l_{max}} m_{\mathbf{x}j}}$, luego la entropía de **x** es:

$$\mathrm{H}(\mathbf{x}) = -\sum_{j=1}^{l_{max}} \frac{m_{\mathbf{x}j}}{\sum_{j=1}^{l_{max}} m_{\mathbf{x}j}} \log_2 \left(\frac{m_{\mathbf{x}j}}{\sum_{j=1}^{l_{max}} m_{\mathbf{x}j}} \right).$$

De forma análoga, se define la *entropía de la posición* j, $\mathrm{H}(j)$, como:

$$\mathrm{H}(j) := -\sum_{\mathbf{y}\in\Sigma} \tilde{P}(\mathbf{y}, j) \cdot \log_2(\tilde{P}(\mathbf{y}, j)).$$

Donde $\tilde{P}(\mathbf{y}, j)$ representa la probabilidad de que en la posición j aparezca el símbolo **y**, observe igualmente que la probabilidad $\tilde{P}(\mathbf{y}, j)$ esta dada por la expresión $\tilde{P}(\mathbf{y}, j) = \frac{m_{\mathbf{y}j}}{\sum_{\mathbf{y}\in\Sigma} m_{\mathbf{y}j}}$, por tanto la entropía quedará escrita como

$$\mathrm{H}(j) = -\sum_{\mathbf{y}\in\Sigma} \frac{m_{\mathbf{y}j}}{\sum_{\mathbf{y}\in\Sigma} m_{\mathbf{y}j}} \log_2 \left(\frac{m_{\mathbf{y}j}}{\sum_{\mathbf{y}\in\Sigma} m_{\mathbf{y}j}} \right).$$

La Figura 2.1 ilustra la nube secuencias y la matriz combinatoria M para $S = \{aa, ab\}$ con un gap, entonces $P(\mathbf{a}, 1)$ es igual a $4/9$ mientras que $\tilde{P}(\mathbf{a}, 1) = 1$, por tanto $\mathrm{H}(1) = 0$ y $\mathrm{H}(\mathbf{a}) = 1{,}53$.

1	2	3
a	a	
a		a
	a	a
a	b	
a		b
	a	b

	1	2	3
a	4	3	2
b	0	1	2

FIGURA 2.1. *Nube de secuencias y Matriz combinatoria.*

3. Algoritmo. A continuación se describe un algoritmo heurístico que genera una aproximación a la subsecuencia común más larga de una colección de secuencias S.

3.1. Alineamiento de una secuencia s con una secuencia patrón p. En esta sección se describirá en detalle el proceso de alineamiento de las secuencias patrón con las secuencias de entrada, estas determinan un conjunto de subsecuencias que son consideradas para la aproximación a la solución de la subsecuencia común más larga de S. Para ello, cada secuencia s en S es alineada con cada secuencia patrón p en P, el resultado de cada alineamiento será considerado para estimar las aproximaciones finales al problema.

El proceso de alineamiento usa como criterio el valor de la entropía de cada símbolo en Σ para definir el orden en el que se alinean los símbolos de cada secuencia, posteriormente se usa el criterio de la posición con menor entropía para alinear cada símbolo.

Dada una secuencia $s \in S$ y una secuencia patrón $p \in P$, se describe el proceso de alineación de s con p de forma general y a su vez se ilustra cada uno de los pasos con un ejemplo.

Paso 1. Inicialmente se construye una matriz llamada la matriz de alineamiento de s con p, cuyas dimensiones son, dos filas y l_{max} columnas. Donde la primera fila contiene los caracteres de la secuencia patrón, y la segunda fila contendrá números enteros; los valores iniciales de la segunda fila son ceros. De ahora en adelante esta matriz se denota por $A = [a_{ij}]$, los valores de sus entradas se irán modificando a medida que se completa el alineamiento.

Para empezar, es necesario definir algunas funciones adicionales para el algoritmo. Primero, la función *primerBloque* toma una lista de número enteros y devuelve el primer conjunto de números consecutivos de la lista.

La función $bloque(\sigma, h)$ devuelve el primer bloque de columnas mayores o iguales a h de la matriz A, tales que contengan al símbolo σ en la primera fila y cero en la segunda. Es decir,

$$bloque(\sigma, h) = primerBloque(\{t : 1 \leq t \leq l_{max}, a_{1t} = \sigma, a_{2t} = 0, t \geq h\})$$

Paso 2. Se ordenan los símbolos del alfabeto Σ de mayor a menor según su valor de entropía, este alfabeto ordenado se denota con Σ_0, el cual define el orden en que los caracteres de s son alineados con p.

Paso 3. Se toma el primer elemento $\sigma \in \Sigma_0$ y se selecciona las posiciones en donde aparece el símbolo σ en s, a esta lista la denotamos por $C = \{c_1, c_2, \cdots, c_k\}$ donde C esta ordenado de forma creciente. En caso de que C sea vacío, se aplica el paso 4, de otro modo se define d como $d := (d_1, d_2, \cdots, d_k)$, con valor inicial $d = (0, \ldots, 0)$.

Para cada c_i en C (recorrido en orden ascendente), se aplica alguno de los siguientes casos:

1. Si $\{t : d_t \neq 0\} = \emptyset$ y $bloque(\sigma, 1) = \emptyset$, se continúa el algoritmo sin alinear símbolo.
2. Si $\{t : d_t \neq 0\} = \emptyset$ y $bloque(\sigma, 1) \neq \emptyset$, se define j_i como el menor número $j_i \in bloque(\sigma, 1)$ tal que

$$\mathrm{H}(j_i) = \min_{t \in bloque(\sigma,1)} \mathrm{H}(t)$$

   Luego, si se considera $a_{2j_i} = c_i$, el conjunto $\{a_{2t} : 1 \leq t \leq l_{max}, a_{2t} \neq 0\}$ está en orden creciente, entonces se actualiza la entrada a_{2j_i} de la matriz A con c_i y $d_i = c_i$. De otro modo, $a_{2j_i} = 0$ y $d_i = 0$.
3. Si $\{t : d_t \neq 0\} \neq \emptyset$ y $bloque(\sigma, 1) = \emptyset$, se continúa el algoritmo sin alinear símbolo.
4. Si $\{t : d_t \neq 0\} \neq \emptyset$ y $bloque(\sigma, 1) \neq \emptyset$, sea $i^* := max\{t : d_t \neq 0\}$.
   Se debe analizar el conjunto $bloque(\sigma, j_{i^*} + c_i - c_{i^*})$, si éste es vacío, se continúa el algoritmo sin alinear símbolo, de lo contrario se define j_i como el menor $j_i \in bloque(\sigma, j_{i^*} + c_i - c_{i^*})$ tal que

$$\mathrm{H}(j_i) = \min_{t \in bloque(\sigma, j_{i^*} + c_i - c_{i^*})} \mathrm{H}(t)$$

   Luego, si se considera $a_{2j_i} = c_i$, el conjunto $\{a_{2t} : 1 \leq t \leq l_{max}, a_{2t} \neq 0\}$ está en orden creciente, entonces se actualiza la entrada a_{2j_i} de la matriz A con c_i y $d_i = c_i$. De otro modo, $a_{2j_i} = 0$ y $d_i = 0$.

Paso 4. Repetir el paso 3 tomando Σ_0 como $\Sigma_0 - \sigma$. Este procedimiento se realiza hasta que Σ_0 sea vacío.

Paso 5. Por último, se cambia cada número no nulo de la segunda fila de A por el caracter correspondiente a la columna, es decir, $a_{2t} = a_{1t}$, mientras que las posiciones nulas se dejan en blanco. Por lo anterior, la secuencia final resultante del alineamiento de s con p es la secuencia de caracteres que aparece en la segunda fila de A.

Ejemplo de alineación de s con p. Sea $S = \{abab, bcca, cab\}$ y $g = 2$, $l_{max} = 6$. El desarrollo del algoritmo para la alineación de $s = abab$ con una secuencia patrón $p = accaab$ es el siguiente.

- Paso 1.
  La matriz inicial A de alineamiento de s con p es:

a	**c**	**c**	**a**	**a**	**b**
0	0	0	0	0	0

- Paso 2.
  El alfabeto de S es $\Sigma = \{a, b, c\}$, teniendo en cuenta la Tabla 3.1 que muestra la entropía de cada símbolo, se concluye que $\Sigma_0 = \{a, b, c\}$ es decir, se alinean primero todas las $a's$, luego las $b's$ y por último las $c's$.

	1	**2**	**3**	**4**	**5**	**6**	$\mathrm{H}(\sigma)$
a	10	8	10	13	14	10	2.568
b	10	10	8	7	10	20	2.488
c	10	12	12	10	6	0	2.284
$\mathrm{H}(j)$	1.585	1.566	1.566	1.541	1.506	0.918	

CUADRO 3.1
Matriz Combinatoria con Entropías

- Paso 3. El primer símbolo de Σ_0 es $\sigma = a$. El conjunto de posiciones donde aparece a en s es $C = \{1, 3\}$ y $d = (0, 0)$. Para cada c_i en C se aplican los casos del 1 al 4 según corresponda:
  Para $c_1 = 1$, note que $\{t : d_t \neq 0\} = \emptyset$ y $bloque(a, 1) = \{1\}$, por consiguiente se aplica el caso 2, luego $j_1 \in \{1\}$ así que, $j_1 = 1$ y $a_{2_{j_1}} = a_{21}$. Suponiendo que $a_{21} = 1$ entonces, el conjunto $\{a_{2t} : 1 \leq t \leq l_{max}, a_{2t} \neq 0\}$ es creciente, así $d = (1, 0)$ y se escribe $a_{21} = 1$ en la matriz A:

a	**c**	**c**	**a**	**a**	**b**
1	0	0	0	0	0

  Para $c_2 = 3$, se ve que $\{t : d_t \neq 0\} = \{1\}$ y $bloque(a, 1) = \{4, 5\}$, por lo tanto se aplica el caso 4.
  Como $i^* = max\{t : d_t \neq 0\} = 1$, entonces $bloque(a, j_{i^*} + c_i - c_{i^*}) = bloque(a, 3) = \{4, 5\}$, luego $j_2 = 5$ por ser la posición con menor entropía.
  Si suponemos que $a_{25} = 3$, el conjunto $\{a_{2t} : 1 \leq t \leq l_{max}, a_{2t} \neq 0\}$ está en orden creciente, así $d = (1, 3)$ y se escribe $a_{25} = 3$ en la matriz A:

a	**c**	**c**	**a**	**a**	**b**
1	0	0	0	3	0

- Paso 4. Regresa al Paso 3 con $\Sigma_0 = \{b, c\}$.
- Paso 3. Ahora el primer símbolo de Σ_0 es $\sigma = b$. El conjunto de posiciones en donde aparece b en s es $C = \{2, 4\}$, se inicia $d = (0, 0)$.
  Para cada c_i en C se aplican los casos del 1 al 4 según corresponda:
  Para $c_1 = 2$ note que $\{t : d_t \neq 0\} = \emptyset$ y $bloque(b, 1) = \{6\}$ por lo tanto, se aplica el caso 2. Entonces, $j_1 \in \{6\}$, así que $j_1 = 6$. Se supone que $a_{26} = 2$, así pues el conjunto $\{a_{2t} : 1 \leq t \leq l_{max}, a_{2t} \neq 0\} = \{1, 3, 2\}$ no está en orden creciente, en consecuencia $d = (0, 0)$ y la matriz A no se modifica:

a	**c**	**c**	**a**	**a**	**b**
1	0	0	0	3	0

  Para $c_2 = 4$, se ve que $\{t : d_t \neq 0\} = \emptyset$ y $bloque(b, 1) = \{6\}$, luego se aplica caso 2. Entonces $j_2 \in \{6\}$ así que $j_2 = 6$. Se supone que $a_{26} = 4$, por tanto, el conjunto $\{a_{2t} : 1 \leq t \leq l_{max}, a_{2t} \neq 0\} = \{1, 3, 4\}$ es creciente, en consecuencia $d = (0, 4)$ y se escribe $a_{26} = 4$ en la matriz A:

a	**c**	**c**	**a**	**a**	**b**
1	0	0	0	3	4

- Paso 4. Vuelve al Paso 3 con $\Sigma_0 = \{c\}$.
- Paso 3. El primer símbolo de Σ_0 es $\sigma = c$. El conjunto de posiciones donde aparece c en s es vacío, luego vuelve al Paso 4.
- Paso 4. Ahora $\Sigma_0 = \emptyset$, entonces vamos al Paso 5.
- Paso 5. Se cambia la segunda fila de la matriz A por sus respectivos símbolos:

a	**c**	**c**	**a**	**a**	**b**
a				a	b

La subsecuencia común obtenida del alineamiento es aab.

3.2. Alineamiento de S con P. En la sección anterior se describió el procedimiento para alinear una secuencia s de S con una secuencia patrón p de P. En esta sección se realiza el alineamiento de cada una de las secuencias de S con cada una de las secuencias de P. Es decir, para cada s de $S = \{s_1, \ldots, s_n\}$ y cada secuencia patrón p de $P = \{p_1, \ldots, p_w\}$, se realiza el alineamiento como se indicó anteriormente. Los resultados de estos alineamientos se presentan dentro de un conjunto de matrices, de tal forma que la matriz k-ésima contiene en su i-ésima fila la secuencia final del alineamiento de s_i con p_k; se denota con U al conjunto de todas las secuencias resultantes de los alineamientos de s_i con p_k, $1 \leq i \leq n$ y $1 \leq k \leq w$.

Por ejemplo, para $S = \{abab, bcca, cab\}$ con $g = 2$, el conjunto de secuencias patrón es $P = \{accaab, cccaab, bccaab\}$. Entonces los alineamientos resultantes se muestran en las matrices de la Tabla 3.2, luego el conjunto de las secuencias resultantes de los alineamientos es $U = \{aab, a, ab, cca, cab, bcca, ca\}$.

a	**c**	**c**	**a**	**a**	**b**
a				a	b
a					
a					b

c	**c**	**c**	**a**	**a**	**b**
				a	b
	c	c		a	
	c			a	b

b	**c**	**c**	**a**	**a**	**b**
				a	
b	c	c		a	
	c			a	

CUADRO 3.2
Matrices de alineamiento de S con P

4. Aproximación Final. Finalmente, el algoritmo escoge como mejor aproximación al problema de la Subsecuencia Común más Larga de S al subconjunto F de secuencias de U que son subsecuencias comunes de S y tienen longitud máxima, formalmente, $F = \{\bar{s} \in U | \bar{s} \in SC(S), \forall \tilde{s} \in SC(S)(\tilde{s} \in U \rightarrow |\tilde{s}| \leq |\bar{s}|)\}$, donde $SC(S)$ representa el conjunto de todas subsecuencias comunes de S.
Para $S = \{abab, bcca, cab\}$ y $g = 2$, el conjunto resultante del alineamiento de S con P es $U = \{aab, a, ab, cca, cab, bcca, ca\}$, de las cuales tan solo la secuencia a es subsecuencia común de S, entonces el conjunto final de aproximaciones al problema de la subsecuencia común más larga de S generado por el algoritmo es $F = \{a\}$. Observe que en efecto la secuencia a es una solución al problema, sin embargo, no es la única solución, ya que b también lo es.

5. Comentarios finales. El algoritmo descrito en este artículo utiliza la entropía de Shannon con el propósito de capturar la información común que tienen los símbolos distribuidos en una colección de secuencias con un cierto número de gaps. Cabe resaltar que, aunque la solución sea vacía, el conjunto F contiene información relevante de la solución del problema.

Como ya se mencionó, no se conocen registros de que este método haya sido usado en estudios previos, por tanto confiamos en que las ideas presentadas en este artículo sirvan para generar nuevos algoritmos heurísticos que puedan alcanzar mejores aproximaciones a la solución del problema.

ORCID and License
Ross Mary Sáenz https://orcid.org/0000-0002-3147-6752.

Referencias

[1] Blin, G., Bulteau, L., Jiang, M., Tejada, P. J. And Vialette, S. Hardness of Longest Common Subsequence for Sequences with Bounded Run-Lengths. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012; 138–148.
[2] Cover, T. M., and Thomas, J. A. Elements of information theory. John Wiley & Sons, 2012.

[3] Forero, W. E. S. Estudio comparativo de algoritmos para el problema de la subsecuencia común más larga restringida / comparative study of algorithms for the constrained longest common subsequence problem. Magíster en ciencias de la Ingeniería de sistemas y computación, Junio 2010.

[4] Hakata, K., and Imai, H. The longest common subsequence problem for small alphabet size between many strings. Algorithms and Computation (1992), 469–478.

[5] Huang, K., Yang, C.-B., Tseng, K.-T., et al. Fast algorithms for finding the common subsequence of multiple sequences. In Proceedings of the International Computer Symposium, 2004; 1006–1011.

[6] Korkin, D. A new dominant point-based parallel algorithm for multiple longest common subsequence problem. Technical Report TR01-148, Univ. of New Brunswick, Tech. Rep. 2001.

[7] Li, Y., Li, H., Duan, T., Wang, S., Wang, Z., and Cheng, Y. A real linear and parallel multiple longest common subsequences (mlcs) algorithm. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016;, ACM:1725–1734.

[8] Maier, D. The complexity of some problems on subsequences and supersequences. Journal of the ACM (JACM), 1978; 25(2), 322–336.