Notas sobre complejidad e implementación del algoritmo UPGMA para árboles ultramétricos *

Abdiel Emilio Cáceres González †
Universidad Juárez Autónoma de Tabasco, DACB

Se muestra un análisis sobre la complejidad del algoritmo UPGMA que es utilizado frecuentemente en bioinformática, en particular en el desarrollo de árboles filogenéticos, además se ofrecen notas acerca de su implementación. Este estudio es útil para los analistas y desarrolladores de software especializado en bioinformática, pues ofrece pautas que pueden servir para el análisis de otros algoritmos similares.

Shows an analysis of the complexity of the UPGMA algorithm that is often used in bioinformatics, particularly in the development of phylogenetic trees, we also provide notes on their implementation. This study is useful for analysts and software developers specializing in bioinformatics, it provides guidelines that can be used for the analysis of other similar algorithms.

Palabras claves: UPGMA, Bioinformática, Complejidad, Algoritmos.
Keywords: UPGMA, Bioinformatics, Complexity, Algorithms.

1. Introducción

El primer reto computacional en el campo de la biología fue la taxonomía y una de las soluciones propuestas fue la implementación de árboles filogenéticos [3]. La taxonomía es la ciencia de la clasificación. En este artículo se revisará la complejidad del algoritmo UPGMA[1] y se describirá una mejora al algoritmo que hace ligeramente más eficiente el algoritmo en la práctica [1].

2. Árboles ultramétricos

Árboles filogenéticos, cladogramas, y aún *árboles de vida* son términos que se utilizan para el mismo objeto [10, 7]. Sin embargo, en ciencias computacionales, existe una estructura de datos que tiene las propiedades de los árboles filogenéticos, esta estructura de datos se llama **árbol ultramétrico** [1], definida del siguiente modo:

Definición 1 (Árbol ultramétrico) Sea $A = \{a_1, a_2, \ldots a_n\}$ un conjunto de n individuos (particularmente caracterizados por una secuencia de ADN o de proteínas, etc.). Un árbol ponderado $T = (V, E, d)$ con raíz r y una función para la ponderación de aristas $d : E \to \mathbb{R}^{\geq 0}$ es un **árbol ultramétrico** para el conjunto A, si satisface las siguientes condiciones:

*Recibido el 10 de julio de 2009 y aceptado el 03 de octubre de 2009
†**Dirección postal**: Carr. Cunduacán-Jalpa Km 1, Cunduacán Tabasco, México. A.P. 24 C.P. 86690. Tel.(+52)914 336-0928. **Correo electrónico**: abdielc@acm.org
[1]Unweighted Pair Group Method with Arithmetic-Mean.

1. T es un árbol binario.
2. T tiene exactamente n hojas, etiquetadas con cada uno de los elementos de A.
3. La suma de los pesos de las aritas de cualquier camino desde la raíz hasta cualquier hoja es la misma.

La distancia entre dos vértices arbitrarios x y y de T es la suma de los pesos de las aristas en el camino desde x hasta y; denotamos esta distancia por $\text{dist}_T(x, y)$.

Esta clase de estructura de datos recibe este nombre porque está definida una función de distancia entre todos los elementos del árbol (los vértices), y además se preserva la regla de los tres puntos, la cual establece que para cualesquiera tres elementos $a, b, c \in V$, se cumple que dos de las tres distancias a pares son iguales y no mayores que la tercera [1].

3. Complejidad del algoritmo UPGMA

UPGMA es un algoritmo para crear árboles ultramétricos y que es utilizado en la clasificación de objetos de manera jerárquica, y que es muy frecuentemente utilizado en bioinformática para hacer taxonomías con datos numéricos obtenidos de un conjunto de seres vivos [8]. Un requisito previo para la implementación del UPGMA, es tener un espacio métrico con los elementos a clasificar.

El algoritmo 1 muestra el algoritmo UPGMA tal como se puede leer en [1], donde se menciona que la complejidad de este algoritmo es $O(n^3)$, sin embargo en [5], se ofrece una implementación que reduce la complejidad hasta $O(n^2)$ haciendo uso de suposiciones con respecto a la distancia de los elementos C_1 y C_2, que solamente requieren considerarse de manera local y no global. En este artículo mostraremos paso a paso la complejidad del algoritmo en su versión más popular, sin el uso de pre-condiciones, como se puede observar en el algoritmo 1.

Este algoritmo ha sido implementado en un lenguaje de programación funcional[2], de modo que en el estudio de la complejidad, también se describen notas acerca de su implementación.

3.1 Complejidad línea a línea

En esta sección se ofrecen algunas notas acerca de la complejidad computacional del algoritmo y algunas implicaciones en la implementación. En la literatura ya existen diversos análisis de este algoritmo, sin embargo no ofrecen notas sobre la implementación que es de utilidad para las personas que deseen implementar este algoritmo.

Línea Requiere: Para desarrollar este algoritmo se requieren dos elementos:

1. El conjunto de individuos a clasificar, identificados por alguna característica que permita medir la diferencia entre ellos, en la implementación hecha, se agruparon

[2]Se utilizó `NetLogo v4.1b3` para aprovechar las facilidades en el desarrollo de gráficas.

Algoritmo 1 Algoritmo UPGMA para construir un árbol ultramétrico

Requiere: Un conjunto $A = \{a_1, a_2, \ldots, a_n\}$ de individuos a clasificar;
una función de distancia $\delta : A \times A \to \mathbb{R}^{\geq 0}$

Devuelve: El árbol ultramétrico $T = (V, E, \mathrm{d})$ para A.

1. $\Gamma \leftarrow \{\{a_1\}, \{a_2\}, \ldots, \{a_n\}\}$
2. $\mathrm{dist}_T(\{a_i\}, \{a_j\}) \leftarrow \delta(a_i, a_j)$ con $i, j \in \{1 \ldots n\}$
3. $\mathrm{altura}(\{a_i\}) \leftarrow 0$ para todos los $i \in \{1 \ldots n\}$
4. $V \leftarrow \Gamma$ y $E \leftarrow \emptyset$
5. **while** $|\Gamma| \geq 2$ **do**
   (a) Encontrar $C_1, C_2 \in \Gamma$, $C_1 \neq C_2$, tales que $\mathrm{dist}_T(C_1, C_2)$ es minimal, y hacer $D \leftarrow C_1 \cup C_2$
   (b) $\Gamma \leftarrow (\Gamma - \{C_1, C_2\}) \cup \{D\}$.
   (c) $\mathrm{dist}_T(D, X) = \mathrm{dist}_T(X, D) \leftarrow \dfrac{\mathrm{dist}_T(C_1, X) + \mathrm{dist}_T(C_2, X)}{2}$; $\forall\, X \in \Gamma$.
   (d) $V \leftarrow V \cup \{D\}$
   (e) $E \leftarrow E \cup \{(D, C_1), (D, C_2)\}$
   (f) $\mathrm{altura}(D) \leftarrow \dfrac{\mathrm{dist}_T(C_1, C_2)}{2}$.
   (g) $\mathrm{d}(D, C_1) \leftarrow \mathrm{altura}(D) - \mathrm{altura}(C_1)$ y $\mathrm{d}(D, C_2) \leftarrow \mathrm{altura}(D) - \mathrm{altura}(C_2)$.
6. **end while**

individuos caracterizados por una secuencia de símbolo de ADN. En la implementación del algoritmo se requiere que los conjuntos se traten como listas con elementos sin repetir, a menos que el lenguaje ofrezca primitivas para el manejo de conjuntos.

2. Una función de distancia sobre los individuos a clasificar. La redacción del algoritmo sugiere que la función sea pasada como parámetro, sin embargo es posible que la declaración y la definición de la distancia no sea pasada como parámetro sino que sea accesada desde un ámbito global.

Línea Devuelve: El algoritmo devuelve una lista (V, E, d):

$$(\mathtt{V} = (\mathtt{v}_1, \ldots, \mathtt{v}_{|\mathtt{V}|}), \mathtt{E} = ((\mathtt{v}_{\mathtt{i}_1}, \mathtt{v}_{\mathtt{j}_2}) \ldots), \mathtt{d} = ((\mathtt{e}_1, \mathtt{d}_1), \ldots))$$

Línea 1: El conjunto Γ se inicializa con n subconjuntos unitarios, cada uno de ellos conteniendo el taxón[3] a_i con $1 \leq i \leq n$. Este proceso es de orden $O(n)$, aunque de manera estricta se debe hacer primero la creación de cada conjunto y luego la inserción al conjunto de conjuntos para finalmente hacer la asignación.

Línea 2: Esta línea requiere un tiempo de orden $O(n^2)$ pues se debe recorrer una tabla de orden $n \times n$ para asignar a cada elemento de la tabla la distancia entre dos elementos. En la implementación, la función es un conjunto de la forma:

$$\mathrm{dist}_T = \{(\{a_1\}, \{a_j\}, \delta(a_i, a_j)), \ldots\},$$

[3]Los individuos sujetos a la taxonomía conforman una *taxa*, cuyo singular es *taxón*. Cuando se refiere al conjunto de individuos se dice *taxa*, sin embargo, cuando se refiere a todos los individuos se puede decir *taxones*. Estos términos provienen del latín y se utilizan en biología.

donde cada elemento de la lista es una triada, donde los las dos primeras entradas son los operandos y la tercera entrada es la evaluación de la distancia entre esos operandos.

Para el cálculo de la distancia se utilizó la distancia de Levenshtein [6, 2], porque se mide la distancia en base a su descripción genética como secuencias de símbolos en el alfabeto $\{\mathtt{T}, \mathtt{C}, \mathtt{G}, \mathtt{A}\}$. Este procedimiento hace muy lento el desarrollo del UPGMA, pues el algoritmo para calcular la distancia entre secuencias genéticas es de orden $O(n^2)$, donde ahora n es la longitud de la secuencia más grande de las dos secuencias que se miden. Esto eleva la complejidad a $O(n^3)$ cuando se calcula la distancia *al vuelo*; sin embargo si en un preproceso se hace una lista con las distancias, el acceso a la distancia es de orden $O(1)$ y la complejidad de la línea 2 permanece en $O(n^2)$.

Línea 3: La asignación de 0 para cada elemento de Γ es $O(n)$. En la implementación, altura es una lista de pares, nuevamente la función es tratada como un conjunto de pares, donde la primera entrada es un elemento de Γ y la segunda entrada de cada par es 0 para esta línea.

Línea 4: Las asignaciones tienen un costo constante $O(1)$.

Línea 5: El ciclo **while** se hace n veces, pues la cardinalidad del conjunto Γ disminuye en una unidad en cada iteración, más precisamente, disminuye en dos unidades y aumenta una (ver línea 5b).

Línea 5a: Esta línea tiene un orden de complejidad de $O(n \log n)$ sobre los elementos de la lista dist_T. Para cumplir con los requerimientos del algoritmo, se deben hacer cuatro tareas:

1. Ordenar la lista dist_T en orden ascendente con respecto al tercer elemento de cada triada (la distancia). Esto es $O(n \log n)$ cuando se utiliza un algoritmo de ordenamiento como el *Quick Sort* [9], y $o(n^2 \log n)$ [no $O(n^2 \log n)$] en todo el ciclo **while**. Nóte que esta complejidad es estrictamente menor que $n^2 \log n$, porque en cada iteración del ciclo **while**, se ordena una lista de menor longitud, lo que implica la suma:

$$\begin{aligned} & n \log n + (n-1)\log(n-1) + \cdots + 2 \log 2 \\ &= \sum_{k=2}^{n} k \log k < \sum_{k=2}^{n} k \log n \end{aligned}$$

   En [1] se menciona que este paso agrega una complejidad de $O(n^3)$ porque tiene que realizar la búsqueda del minimal en una matriz cuadrada, esto es un orden de $O(n^2)$ y $O(n^3)$ en todo el ciclo.
2. Encontrar la primera triada $(a, b, \delta(a, b))$ en dist_T, que cumpla $a \neq b$. Esto es $O(1)$, recordemos que ahora la lista ya esta ordenada en forma creciente, y al inicio de la lista se encuentran las que tienen menor distancia.
3. Asignar $C_1 \leftarrow a$ y $C_2 \leftarrow b$. Esto es $O(1)$.
4. C_1 y C_2 son conjuntos disjuntos, pues representan diferentes clados[4]. Como $C_1 \cap C_2 = \emptyset$, para hacer la unión, es suficiente agregar los elementos de C_2 a C_1, esto es de orden $O(n)$, si n es $|A|$ en todo el ciclo **while**, aunque en los casos promedio es de orden $O(\log n)$ porque en cada iteración se agrega un conjunto del doble de tamaño en promedio.

Línea 5b: El conjunto Γ se actualiza retirando dos elementos y agregando uno; los dos elementos retirados se conservan en la unión de ellos, que posteriormente se agrega a Γ como un elemento compuesto. Este paso abona una complejidad de $O(n)$ en todo el ciclo **while**.

[4]Un *clado* se puede ver como un subárbol.

Línea 5c: Esta línea representa la actualización de la función de distancia ultramétrica dist_T, como las distancias son simétricas, entonces es suficiente hacer el cálculo en la mitad de la matriz de $n \times n$, y hacer una simple asignación en la otra mitad de la matriz. Desafortunadamente, hacer esto no representa cambio alguno en la complejidad que sigue siendo de $O(n^2)$, veamos, en la primera iteración, el cálculo se realiza n veces, cuando $|\Gamma| = n$, en la segunda iteración, Γ tiene un elemento menos, de modo que se realiza $n-1$ veces, y así en adelante hasta que $|\Gamma| = 2$, así que el cálculo de los nuevos valores para dist_T es de $n + (n-1) + \cdots + 2 = O(n^2)$ en todo el ciclo **while**.

Línea 5d: Agregar el nuevo vértice al conjunto V de vértices es constante y se realiza n veces, de modo que la complejidad es $O(n)$.

Línea 5e: También es $O(n)$.

Línea 5f: Aunque se tienen que hacer dos operaciones (obtener la distancia ultramétrica dist_T y la división), la complejidad es $O(n)$. Este es el mejor momento para actualizar la función dist_T, ya que hay que quitar todas las triadas que contengan cualquiera de C_1 o C_2, recordemos que se han eliminado de Γ para incluir $C_1 \cup C_2$; esta operación es $O(n)$.

Línea 5g: Finalmente, asignar el peso a las aristas (D, C_1) y (D, C_2) también representa una complejidad $O(n)$.

De modo que concluimos que el algoritmo UPGMA tiene un grado de complejidad $O(n^2 \log n)$, que es menor a lo que se establece en[1], pero igual a lo que se describe en [5] en su versión sin presuposiciones.

4. Actividades involucradas en el análisis de árboles ultramétricos

El resultado del algoritmo UPGMA es un grafo en donde se pueden observar datos útiles en el análisis cladistico[5]. En las figuras 1 y 2 se puede observar un árbol ultramétrico generado por el UPGMA, cuando se inició con una taxa de tamaño 10.

Hay varias actividades que se requieren realizar para obtener un resultado como el de las figuras 1 y 2:

1. **Colección de datos:** Los datos que se requieren son las descripciones de los individuos. En el caso de las figuras citadas, cada individuo está identificado por medio de una secuencia genética como:

   ```
   "gttaatgtagcttaattaataaagcaaggcactgaaaatgccaagatgagtcgcacgact"
   "ccataaacataaaggtttggtcctagccttcctattagtttttagtagacttacacatgc"
   "aaatttcgtgccagccaccgcggtcatacgattaacccaaactaataggcctacggcgta"
   "aagcgtgttcaagatacttttacactaaagttaaaacttaactaagccgtaaaaagctac"...
   ```

   La longitud de cada secuencia puede ser diferente, incluso el genoma completo de cada taxón, sin embargo el tiempo de ejecución para encontrar la diferencia entre dos secuencias es de orden $O(nm)$, lo que afecta considerablemente el desempeño del UPGMA. No se requieren características fenotípicas, pues la clasificación se hace en base a sus secuencias genómicas.

[5]La cladística es la parte de la biología que estudia la clasificación de los organismos vivos.

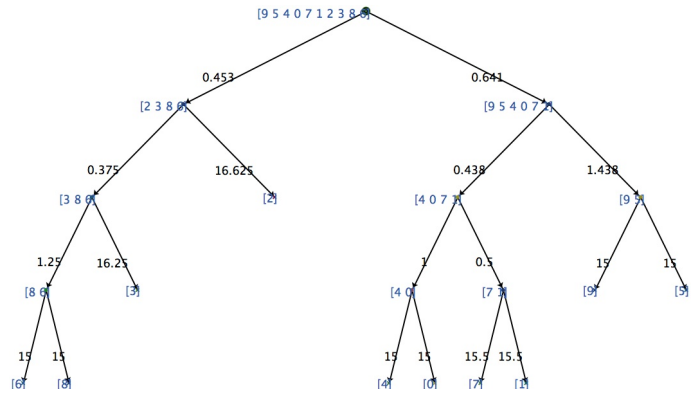

Figura 1. Árbol ultramétrico para una taxa de tamaño 10. Se muestran las aristas etiquetadas con la distancia entre nodos. Cada nodo hoja es un taxón, cada nodo interno es la raíz de un subárbol que representa un clado, y se etiqueta con el índice de la taxa que incluye.

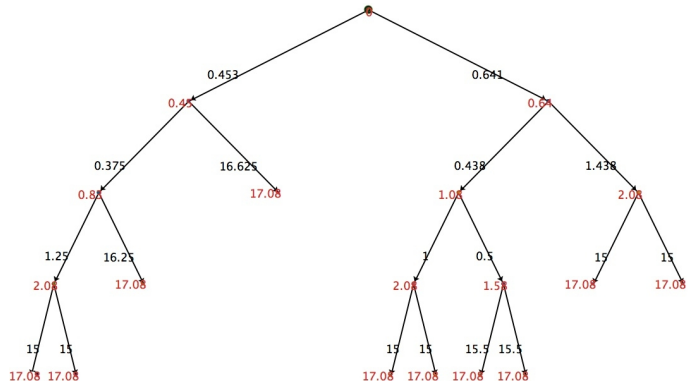

Figura 2. Se muestra el mismo árbol ultramétrico de la figura 1. Cada nodo se ha etiquetado con la distancia desde la raíz del árbol. Nótese que la distancia desde la raíz a cualquier hoja es la misma.

2. **Criterio de distancia:** La distancia empleada en estos ejemplos es la distancia de Levenshtein, que se puede definir como el mínimo número de operaciones de edición requeridas para transformar una secuencia de caracteres en otra. Una descripción e implementación en distintos lenguajes de programación se puede obtener en [4].

3. **Construcción del árbol filogenético:** El UPGMA efectivamente construye un árbol filogenético, sin embargo no es el único método, existen otros métodos como el de la máxima parsimonia, que determina un árbol con menor altura[6]; y el método de la máxima probabilidad, utilizado frecuentemente en análisis estadísticos.

4. **Evaluación del árbol:** La evaluación que se hace es acerca de la robustez del método. Un método es más robusto si genera el mismo árbol, con un una taxa sin importar el orden de los taxones.

[6]Un ábol con menos altura significa menos eventos de especiación, es decir, que las especies sufren menos cambios genéticos.

5. **Visualización de los resultados:** Es importante ofrecer una vista gráfica donde se observen las características importantes, para eso es muy útil ejecutar un algoritmo de visualización del árbol. Este algoritmo generalmente tiene una complejidad de $O(n)$.

5. Conclusiones

El algoritmo UPGMA es $o(n^2 \log n)$, la implementación implica la elaboración de al menos otros dos algoritmos importantes: el de la obtención de las distancias que es de orden $O(nm)$, donde n y m son las longitudes de las secuencias; y la representación visual que es de orden $O(n)$. Considerando el uso de la distancia de Levenshtein, la complejidad del algoritmo es $O(n^2)$, cuando $n >> |A|$; $n = \max\{\text{long}(ADN_{taxon})\}$ la máxima longitud de ADN de todos los taxones. El árbol generado no es completamente robusto, pues el hecho de seleccionar un par minimal, hace que se puedan generar árboles diferentes para el mismo conjunto de taxa.

Agradecimientos

El autor agradece a las autoridades de PROMEP por financiar el proyecto Promep-20080798.

Referencias

[1] Böckenhauer, H.-J., and Bongartz, D. Algorithmic aspects of bioinformatics, 2007.

[2] Cáceres González, A. E. La métrica de levenshtein. *Revista de Ciencias Básicas UJAT 7*, 2 (Diciembre 2008), p 35–43.

[3] Gibas, C., and Jambeck, P. Developing bioinformatics computer skills, 2001.

[4] Gilleland, M. Levenshtein distance, in three flavors. `Internet resource at http://www.merriampark.com/ld.htm` (2009).

[5] Gronau, I., and Moran, S. Optimal implementations of UPGMA and other common clustering algorithms. *Inf. Process. Lett. 104*, 6 (2007), 205–210.

[6] Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady 10* (1966), 707 – 710.

[7] Shuying, L., Dennis K, P., and Hani, D. Phylogenetic tree construction using markov chain monte carlo. *Journal of the American Statistical Association*, 95 (2000), 493 – 508.

[8] Sokal, R. R., and Sneath, P. H. A. *Principles of numerical taxonomy.* San Francisco, CA: W. H. Freeman, 1963. 359p.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*, 2 ed. MIT Press, 2001.

[10] University of California Museum of Paleontology. Phylogenetic systematics, a.k.a. evolutionary trees. *Understanding Evolution [Webpage at `http://evolution.berkeley.edu/evolibrary/article/0_0_0/phylogenetics_01%`]* (Marzo Last checked on March 2009).