COMPUTACIÓN BASADA EN ADN

Posted on 25 septiembre, 2015 by Gabriel A. Caballero Robledo

La maravilla de la interdisciplina es que permite conectar mundos que normalmente no se ven. A finales de los 80 y principios de los 90 del siglo pasado, Leonard Adleman, ma-temático y experto en cómputo, tenía como una de sus líneas de investigación fenómenos relacionados con el SIDA. Según sus propias palabras, para poder tener más credibilidad entre la comunidad científica del tema decidió estudiar a fondo la biología del virus.

Category: Ciencia

Tag: Ensayo Científico

La maravilla de la interdisciplina es que permite conectar mundos que normalmente no se ven.

A finales de los 80 y principios de los 90 del siglo pasado, Leonard Adleman, matemático y experto en cómputo, tenía como una de sus líneas de investigación fenómenos relacionados con el SIDA. Según sus propias palabras, para poder tener más credibilidad entre la comunidad científica del tema decidió estudiar a fondo la biología del virus. Para eso tuvo que pisar el laboratorio y aprender

biología molecular. Ahí conoció los detalles de cómo el funcionamiento de la vida se basa en procesar información almacenada en la molécula de ADN (Ácido desoxirribonucleico). Pero sus ojos de *computólogo* lo llevaron a ver los mecanismos con los que los seres vivos procesan y almacenan la información genética, como pequeñas computadoras. Y mejor aún, pensó en un ingenioso modo para utilizar esos mecanismos y resolver problemas matemáticos complejos.

> *La molécula de ADN está formada por una serie de moléculas más pequeñas llamadas nucleótidos.*

La molécula de ADN está formada por una serie de moléculas más pequeñas llamadas nucleótidos, unidas en forma de cadena. Hay cuatro diferentes nucleótidos denominados por las letras A, T, C y G, que pueden ordenarse de cualquier manera para formar largas cadenas de ADN, por ejemplo GCTATCGACGT. La información biológica está guardada en este tipo de cadenas de forma análoga a las cadenas de ceros y unos (00110101110011) con las que guarda la información una computadora digital. El mecanismo que los organismos vivos tenemos para leer y usar esa información es a través de pequeñas moléculas que funcionan como nanomáquinas, capaces de actuar sobre las cadenas de ADN. La más importante de esas moléculas es la enzima ADN-Polimerasa. Esta molécula es capaz de leer una cadena de ADN y generar su cadena complementaria basándose en las reglas de apareamiento de nucléotidos de Watson y Crick: el nucléotido C es complementario con el G, mientras que el A lo es con el T. Cuando la ADN-Polimersa encuentra un nucleótido en la cadena original, coloca su complementario en la cadena complementaria. De este modo, la cadena complementaria de la cadena del párrafo anterior sería CGATAGCTGCA.

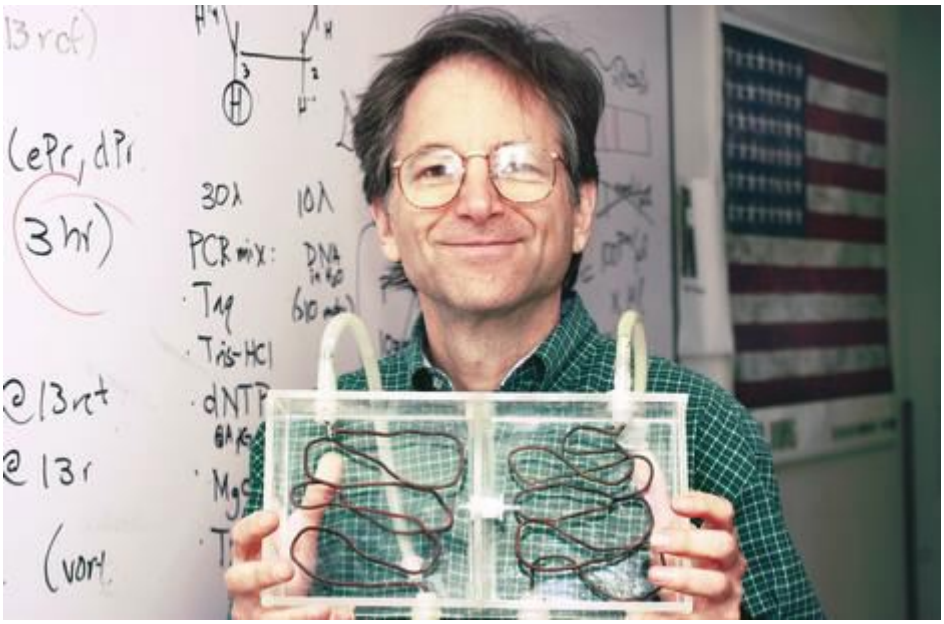
En el sentido más amplio del concepto, una computadora es algo mucho más general que un dispositivo electrónico que sirve para procesar datos, editar documentos y navegar por internet. Cualquier sistema que sea capaz de transformar información de entrada en información de salida aplicando una serie de reglas (un algoritmo) es considerada una computadora. La base de esta definición es la famosa Máquina de Turing: una pequeña máquina con dos brazos. Con uno de ellos lee, uno a uno, los símbolos impresos en una larga banda, y con el otro imprime en otra banda el resultado de aplicar a cada símbolo leído una serie de reglas de procesamiento (un algoritmo) con las cuales ha sido configurada la máquina. La Máquina de Turing es muy simple pero su esencia es la misma que la de cualquier computadora, analógica o digital, por compleja que ésta sea; es una conceptualización muy general de una computadora que por su simpleza y generalidad ha sido muy importante para el estudio teórico de las ciencias computacionales.

> *los avances en la biología molecular todavía no nos permiten diseñar moléculas como la ADN-Polimerasa.*

Cuando Adleman aprendió el modo en que opera la ADN-Polimerasa no pudo verla sino como una

nano-Máquina de Turing, programada para crear cadenas complementarias de ADN. Inmediatamente comenzó a pensar cómo lograr que esa computadora biológica hiciera otro tipo de operaciones computacionales. Sin embargo, los avances en la biología molecular todavía no nos permiten diseñar moléculas como la ADN-Polimerasa para que actúen sobre el ADN con las reglas que nosotros queremos: la ADN-Polimerasa es el resultado de millones de años de evolución. Así es que Adleman tuvo que conformarse con los elementos disponibles en la biología molecular para pensar en alguna forma de hacer cálculos computacionales basados en ADN.

Además de la ADN-Polimerasa hay otras moléculas (nano-máquinas) capaces de operar sobre las cadenas de ADN, como por ejemplo, uniéndolas (la ADN-ligasa) o cortándolas (la ADN-nucleasa). Aunado a esto, la biología molecular ha desarrollado técnicas para crear cualquier cadena de ADN que uno quiera o para separarlas según su tamaño. Echando mano de todos estos elementos, Adleman fue capaz de traducir un problema matemático en cadenas de ADN y resolverlo a través de un algoritmo basado en reacciones bioquímicas e ingeniosas manipulaciones moleculares en el laboratorio. Es decir, inventó la computación basada en ADN.



Leonard Adleman con su computadora basada en ADN

El problema matemático que Adleman decidió resolver con su computadora bioquímica fue una versión especial del problema del viajero, un problema de optimización relacionado con encontrar el camino óptimo para visitar un cierto número de ciudades pasando por cada una de ellas únicamente una vez. El problema del viajero es un problema computacionalmente “muy difícil” y al resolverlo, Adleman no sólo mostró la posibilidad de realizar operaciones computacionales basadas en ADN, sino que puso en evidencia que la naturaleza de su computadora bioquímica es el cómputo

en paralelo, es decir, la realización simultánea de muchas operaciones, en contraste con el cómputo en serie de las computadoras digitales, en donde las operaciones se realizan una después de otra. El cómputo en paralelo potencia por mucho las capacidades de una computadora. La tendencia actual en la computación digital es usar cada vez más el cómputo en paralelo; de ahí el desarrollo de procesadores de varios núcleos o el repunte del cómputo científico basado en tarjetas de video (GPU's).

Pero, ¿qué quiere decir que un problema sea computacionalmente difícil? Las ciencias computacionales clasifican los problemas que pueden resolverse con una computadora de acuerdo al número de operaciones que deben realizarse para obtener la respuesta y cómo crece ese número al aumentar la cantidad de variables del problema: si el número de pasos aumenta polinomialmente se dice que el problema es fácil y se le clasifica como problema tipo P (polinomial). Por otro lado, si la solución a un problema es fácilmente verificable, es decir, el proceso de verificación crece polinomialmente con la cantidad de variables, entonces se le clasifica como problema tipo NP (Non-deterministic polynomial). Por ejemplo, separar por colores un conjunto de piezas Lego es un problema fácil. El algoritmo sería: 1) se toma la pieza; 2) se detecta el color; 3) se coloca en el recipiente del color correspondiente. Si hay X piezas, el problema queda resuelto en 3X pasos. Si hay el doble de piezas, 2X, entonces el número de pasos es también el doble: 3*2X = 6X. Es decir, el número de pasos crece linealmente, que es un polinomio de orden 1. Separar por colores es entonces un problema tipo P, pero también es un problema NP porque verificar que las piezas estén separadas correctamente es fácil. En cambio, armar un rompecabezas es un problema NP, porque una vez que está armado es fácil ver que esté armado correctamente, pero no es un problema tipo P. Para que una computadora pudiera armar un rompecabezas, prácticamente debería probar todas las piezas unas contra otras para saber cuáles embonan correctamente. Si el rompecabezas tiene X piezas el número de pasos necesarios es X factorial, es decir,

X*(X-1)*(X-2)*….*4*3*2*1. Si X = 10, el número de pasos sería 3,628,800; si X = 20 se necesitarían $2.4x10^{18}$ pasos (¡2.4 seguido de 18 ceros!). Con tal tasa de crecimiento del número de pasos necesarios en el algoritmo al aumentar el número de piezas, el problema se vuelve inmanejable en términos prácticos muy rápidamente.

> *¿Será que todo problema NP es también P?*

En 1971, Stephen Cook demostró que todo problema tipo P es también NP. Una vez probado esto, surge de manera natural la pregunta, ¿será que todo problema NP es también P? La pregunta parece ingenua: el problema del rompecabezas es claramente NP, pero no parece ser fácil de armar para una computadora. Así es que la respuesta a la pregunta parece ser un obvio y contundente NO y debería de ser fácil para un matemático demostrarlo formalmente. Pues no es así. El problema P versus NP es el problema teórico sin resolver más importante de las ciencias computacionales. De hecho, es uno de los Siete Problemas del Milenio, por los que el Clay

Mathematics Institute ofrece un millón de dólares, por cada problema, a quien los resuelva.

Mostrar que todo problema P es también NP no es difícil. Lo verdaderamente valioso del trabajo de Cook de 1971 fue que demostró que dentro del conjunto de problemas NP, existe otra clasificación de problemas llamados NP-completos. Este tipo de problemas tienen dos propiedades interesantes: 1) todo problema NP puede ser mapeado o traducido a uno NP-completo, por lo que si alguien encuentra un algoritmo fácil para resolver uno de estos problemas, entonces todo problema NP sería fácil y P sería igual a NP; 2) todos los problemas NP-completos descritos hasta ahora son difíciles, es decir, los algoritmos para resolverlos crecen exponencialmente al aumentar las variables. Cook demostró que todo problema NP puede ser traducido a una "oración lógica" del tipo en donde las letras a, b, c y d puede tomar los valores verdadero o falso y están unidas por los operadores "y", "ó", "no". Si hay una combinación de verdaderos y falsos que haga que la oración total sea verdadera, entonces la oración tiene solución ("satisfiable" en inglés). Si no existe tal combinación, y la oración siempre es falsa, entonces no tiene solución. Encontrar si una oración de este tipo tiene o no solución se le conoce como un problema tipo SAT (abreviación de "satisfiability" en inglés): (a y no b) tiene solución; (a y no a) no tiene solución.

> *Muchos problemas cotidianos son del tipo NP-completo.*

El ejemplo del rompecabezas es un problema NP-completo. Y no es el único. Muchos problemas cotidianos son de este tipo: asignar pilotos y tripulación a aviones de una compañía aérea, distribuir víveres en una zona de desastre, sincronizar semáforos, acomodar la mayor cantidad de cajas de diferentes tamaños en un camión, jugar tetris, y un largo etc. Pero la relevancia de P vs NP no solo radica en lo cotidiano de estos problemas. Mientras no se demuestre que P es diferente de NP existe la posibilidad de que todos los problemas computacionales puedan ser fáciles de resolver para una computadora, lo cual tendría consecuencias enormes: la capacidad de cómputo de nuestras máquinas sería mucho mayor del límite que actualmente parecen tener, además de que pondría a todo el comercio y bancos en graves problemas dado que sus medidas de seguridad están basadas en problemas de criptografía NP-difíciles.

Adleman resolvió con su computadora bioquímica el problema de determinar si es posible visitar siete ciudades conectadas por catorce vuelos directos pasando únicamente una vez por cada ciudad. Como dijimos, este tipo de problemas son difíciles de resolver computacionalmente, pero únicamente cuando se quiere visitar muchas ciudades. Cuando solo son siete, entonces la solución se encuentra rápidamente, incluso sin usar una computadora. Pero Adleman sólo quería demostrar que resolver el problema era posible, y usar más ciudades habría implicado mucho más trabajo en el laboratorio.

> *El trabajo de Adleman motivó a otros investigadores a resolver problemas difíciles usando computadoras moleculares.*

El trabajo de Adleman motivó a otros investigadores a resolver problemas difíciles usando computadoras moleculares. En el año 2000, Q. Liu y colaboradores reportaron haber resuelto el más difícil de los problemas NP, el problema 3-SAT, usando cómputo basado en ADN. En ese trabajo los autores mostraban que podían resolver un problema 3-SAT usando un algoritmo que crece polinomialmente con el número de variables del problema si no se consideran los pasos para crear las cadenas de ADN necesarias al inicio del proceso. ¿Un problema NP-completo resuelto por un algoritmo polinomial? ¿Había demostrado el cómputo basado en ADN que NP = P? Todavía no. El algoritmo de Q. Liu et al. era polinomial si no se consideraban los pasos iniciales de fabricación de las cadenas de ADN, pero no estaba claro cómo evaluar la complejidad de esos pasos iniciales. Los autores sugerían que, aunque en ese momento no había las herramientas necesarias para hacer ese paso inicial lo suficientemente sencillo para resolver problemas de más variables, esas herramientas ya estaban en proceso de desarrollarse y no tardarían mucho en estar listas.

> *El cómputo basado en ADN no es viable para resolver problemas tipo SAT*

Pero no todos compartían su optimismo. En el mismo número de la revista Nature en que fue publicado el trabajo de Q. Liu y colaboradores, M. Ogihara y A. Ray cuestionan que la síntesis inicial de cadenas de ADN sea un proceso fácil en el sentido computacional de la palabra. Y no sólo la síntesis inicial de cadenas de ADN es un problema, también lo es el "ruido" intrínseco del proceso: el material que no logra separarse perfectamente en los pasos del algoritmo o la interacción de cadenas de ADN que no son perfectamente complementarias pero que se parecen mucho, son fuentes de error que, aunque pequeñas, imponen un límite al tamaño de problemas difíciles que pueden resolverse con esta técnica. De hecho, ya se ha demostrado que debido a este ruido, el cómputo basado en ADN no es viable para resolver problemas tipo SAT mejor que una computadora digital.

Pero esto no quiere decir que el cómputo basado en ADN no sea útil. M. Ogihara y A. Ray fueron visionarios no sólo por dudar de la capacidad del cómputo basado en ADN para resolver problemas difíciles, sino también por proponer que el mayor potencial de esta herramienta está en que para ciertas aplicaciones, puede ser muy útil que un dispositivo bioquímico realice operaciones computacionales de manera autónoma sin necesidad de estar conectado a una computadora digital. Por ejemplo, uno pudiera pensar en un dispositivo médico implantado dentro del organismo que pudiera determinar a partir de operaciones computacionales bioquímicas la necesidad de liberar fármacos en función de las necesidades inmediatas del organismo. Actualmente esto todavía no es una realidad, pero hacia allá apunta la investigación y el desarrollo en esta área y ya hay avances importantes. A pesar de sus limitaciones, la computación basada en ADN no deja de tener características que pueden ser muy útiles para desarrollar aplicaciones biomédicas: capacidad de almacenamiento de mucha información de manera muy condensada (la información genética almacenada en las células es la máxima expresión de esa capacidad); su enorme capacidad de

realizar operaciones computacionales de forma simultánea (computación en paralelo); cada operación realizada con ADN requiere menos energía que si se realiza con una computadora convencional (eficiencia energética).

Han pasado veinte años desde el nacimiento del cómputo basado en ADN y todavía la comunidad científica no termina de entender su verdadero potencial y sus limitaciones. El tiempo ha mostrado que la computación basada en ADN no resolverá el problema P vs NP, pero la idea de Adleman no deja de ser genial. Como él mismo señaló en su momento, su experimento se enmarca dentro del florecimiento de un joven campo del conocimiento que une a las ciencias computacionales con la biología molecular y que promete sorprendentes descubrimientos y aplicaciones. Lo mejor está por venir... *a nous de jouer!* C^2

Referencia

En 1973 el ucraniano Leonid Levin llegó a resultados similares de forma independiente.

Lecturas recomendadas:

- Adleman, L. M. Sci. Am 279, 54-61 (1998).
- Liu Q. et al. *Nature* 403, 175-179 (2000).
- Ogihara M. y Ray A. *Nature* 403, 143-144 (2000).
- Pavlus J., *Machines of the Infinity*, Sci. Am. 307, (iPad) (2012).