matemática: Una publicación de FCNM – ESPOL
2013, Vol. 11, No. 2

ALGORITMOS FACTIBLES, PROBLEMAS TRATABLES Y LA COMPLEJIDAD COMPUTACIONAL DE UNA VARIANTE DEL PROBLEMA DE LA DIVERSIDAD MÁXIMA

Sandoya Fernando[1]

Resumen. *La intención de este artículo es dar una demostración formal del carácter NP-duro de una nueva variante del conocido problema de optimización combinatoria de la diversidad máxima, esta nueva variante es denominada problema del máximo promedio. Además se presenta una breve revisión de las nociones y conceptos relacionados con la NP-dureza, y la conjetura P vs.NP, con el fin de comprender la naturaleza de los problemas de optimización, y por qué algunos de ellos se pueden considerar fáciles y otros pueden llamarse difíciles.*

Palabras Claves: complejidad computacional, optimización combinatoria, problema de la diversidad máxima.

Abstract. *The purpose of this article is to give a formal proof of the NP-hard nature of a new variant of the classical maximum diversity problem, this new variant is called average maximum problem. It also presents a brief review of the notions and concepts related to the NP-hardness, and conjecture P vs.NP, in order to understand the nature of the optimization problems, and why some of them can be considered easy and others may call difficult.*

Keywords: Computational complexity, combinatorial optimization, máximum diversity problem.

Recibido: Junio 2013
Aceptado: Agosto 2013

1. INTRODUCCIÓN

Es algo evidente, validado empíricamente, que los problemas de toma de decisiones o de optimización que se abordan en la realidad tienen diferentes grados de "dificultad", hay problemas que son fáciles de resolver, en el sentido de que podemos llegar a su solución óptima con un grado moderado de esfuerzo analítico y computacional, mientras que otros son difíciles y aparentemente imposibles de resolver. ¿Pero esto es formalmente cierto?, es decir podríamos afirmar que ¿existe realmente una diferencia cualitativa entre estos problemas, que hace que los primeros puedan resolverse y los otros no, y que además este hecho nunca va a cambiar por más que evolucionen las matemáticas y la tecnología? La teoría de la complejidad computacional trata de dar una respuesta a esta discusión, en la primera parte de este artículo se abordan los aspectos más importantes. Para un detalle más profundo sobre esta teoría, el lector puede remitirse al influyente libro de texto de M. Garey y D. Johnson "Computers and Intractability: A Guide to the Theory of NP-Completeness" (1). En la segunda parte se da una demostración formal de la NP-dureza de un nuevo modelo de optimización combinatoria.

[1] *Sandoya Fernando, Ph.D. (c), Profesor de la Escuela Superior Politécnica del Litoral (ESPOL).*
(e_mail: fsandoya@espol.edu.ec).

2. ALGORITMOS FACTIBLES Y PROBLEMAS TRATABLES

De manera informal, podemos denominar a los problemas "difíciles" como los problemas duros. Entonces a un problema $\mathcal{P}_0$ se le denomina $NP-duro$ si es al menos tan duro como otros problemas de cierta clase razonable de problemas. En lo que sigue se trata de formalizar esta definición, que se basa en las nociones de algoritmos factibles y problemas tratables.

2.1 ALGORITMOS FACTIBLES

La noción de $NP-dureza$ está relacionada al hecho empíricamente conocido de que algunos algoritmos son factibles y otros no lo son. Donde la factibilidad de un algoritmo está relacionada íntimamente con el tiempo computacional requerido para su ejecución. Por ejemplo, si para alguna entrada x de longitud $len(x) = n$, un algoritmo requiere 2^n pasos, entonces para una entrada de tamaño mediano, digamos $n = 200$, se necesitarán 2^{200} pasos, trabajo que no podría realizarse en un tiempo humanamente razonable, por más que evolucione la velocidad de los computadores.
Formalmente, denominamos *algoritmo de tiempo exponencial* a cualquier algoritmo cuyo tiempo de ejecución $t(n)$ con entradas de longitud n crece al menos como una función exponencial, es decir, para el que:

$$\exists c > 0, N \in \mathbb{N}, \forall n \geq N, t(n) \geq e^{c\,n}$$

Como resultado de esta definición, los algoritmos de tiempo exponencial son usualmente considerados no factibles. Aunque hay que aclarar que el hecho de que un algoritmo sea no factible no significa que nunca puede ser aplicado, simplemente indica que hay casos para los que el tiempo de ejecución será tan grande que no es práctico utilizarlos, aunque para ciertas entradas (de talla pequeña) el algoritmo si podría ser útil.

Por otro lado, el algoritmo se denomina de *tiempo polinomial* si el tiempo de ejecución crece sólo como un polinomio de la talla de las entradas n; es decir, si y solo si:

"Existe un polinomio $P(n)$ tal que para toda entrada x de longitud $len(x)$, el tiempo computacional $t_{\mathcal{U}}(x)$ del algoritmo $\mathcal{U}$ con la entrada x está acotado por $P\big(len(x)\big)$: $t_{\mathcal{U}}(x) \leq P\big(len(x)\big)$".

Como resultado de la definición, estos algoritmos de tiempo polinomial suelen ser bastante factibles.

A partir de estas dos posibilidades, en la literatura sobre complejidad computacional se ha consensuado sobre la siguiente definición de factibilidad: "Un algoritmo $\mathcal{U}$ se denomina factible si y solo si es de tiempo polinomial". En muchos casos prácticos, esta definición describe adecuadamente la idea intuitiva de factibilidad: los algoritmos de tiempo polinomial son usualmente factibles, y los algoritmos de tiempo no polinomial son usualmente no factibles. Sin embargo habrán casos en que esta intuición no funcione, por ejemplo si un algoritmo tiene un tiempo de ejecución $2^{500}n$, es claramente de tiempo polinomial, pero obviamente infactible. Pero a pesar de estos aparentes vacíos en esta formalización, por ahora "tiempo polinomial" es la mejor descripción conocida de factibilidad, por tanto establecemos la siguiente definición:

Definición 1: Un algoritmo $\mathcal{U}$ se denomina factible si y solo si:

"Existe un polinomio $\mathrm{P(n)}$ tal que para toda entrada x de longitud $\mathrm{len(x)}$, el tiempo computacional de ejecución $\mathrm{t}_{\mathcal{U}}\mathrm{(x)}$ del algoritmo $\mathcal{U}$ está acotado por $\mathrm{P\big(len(x)\big)}$".

Donde $len(x)$ representa la longitud de la entrada x, es decir el número de bits que forma esta entrada.

2.2 PROBLEMAS TRATABLES

Una vez que se ha formalizado la definición de algoritmo factible, se puede describir qué es un problema manejable (o tratable) y que es un problema inmanejable (o intratable):

Definición 2: Si existe un algoritmo de tiempo polinomial que resuelve todas las instancias de un problema, este problema se dice que es tratable, caso contrario se dice que es intratable. A estos problemas tratables también se les denomina "fáciles de resolver", o "problemas de la clase P"

Es decir, para decidir sobre el carácter del problema, o bien tenemos explícitamente un algoritmo de tiempo polinomial para resolver todas las instancias del problema, o tenemos una demostración de que no existe un algoritmo de ese tipo.

Desafortunadamente, en muchos casos, no se conoce ni lo uno ni lo otro, o es sumamente complicado llegar a establecer una de las dos cosas. Esto no significa que no haya solución al problema, pues en lugar de la falta de información ideal, tenemos otra información que es casi tan buena. Es decir, en algunos casos, no sabemos si el problema se puede resolver en tiempo polinomial o no, pero por ejemplo, sí podríamos saber que este problema es tan duro como pueden ser algunos problemas prácticos: entonces si pudiéramos solucionar este problema fácilmente, habría un algoritmo que resuelve todos los problemas con facilidad, pero la existencia de tal algoritmo es muy improbable, pues nuestra intuición, y sobretodo experiencia empírica, nos indica que en la realidad hay problemas muy difíciles. A estos problemas "duros" se les denomina intratables.

Para formular esta definición en términos más formales, debemos primero describir lo que entendemos por un *problema práctico* y que significa *reducir* un problema a otro.

Para describir que es un problema práctico, supongamos que:

- Tenemos alguna información, representada por x, y
- Conocemos la relación $R(x,y)$ entre la información conocida x y el objeto deseado y

Suponemos que tanto la información x como el objeto deseado y se han representado como sucesiones binarias, esto siempre se puede hacer ya que en la computadora cualquier cosa puede ser representada como una sucesión binaria.

Por ejemplo si se tiene una proposición matemática x, y el objeto deseado de estudio y es o bien una demostración de que x es verdadero, o una refutación de x; es decir una demostración de que x es falso. Entonces $R(x,y)$ significa que y es una demostración cualquiera de x, o de "no x".

Para un problema que es práctico, debemos tener una manera de comprobar si la solución propuesta es correcta. En otras palabras, debemos suponer que existe un algoritmo factible que chequee la veracidad de $R(x,y)$ dados x e y. Si no existe tal algoritmo factible, entonces no existe un criterio para decidir si hemos logrado encontrar una solución o no al problema.

Otro requerimiento para un problema de la vida real es que en tales problemas, usualmente conocemos una cota superior para la longitud $len(y)$ de la descripción de y. En el ejemplo anterior: una demostración no debería ser demasiado grande, sino sería imposible comprobar si se trata de una prueba o no.

En todos los casos, es necesario que un usuario sea capaz de leer la solución deseada símbolo tras símbolo, y el tiempo requerido para la lectura debe ser factible. Anteriormente se formalizó "tiempo factible" como un tiempo que está acotado por algún polinomio de $len\,(x)$. El tiempo de lectura es proporcional a la longitud $len\,(y)$ de la respuesta y. Por lo tanto, el hecho que el tiempo de lectura es acotado por un polinomio de $len\,(x)$ significa que la longitud de la salida y está también acotada por algún polinomio de $len\,(x)$. Es decir: $len\,(y) \leq P_L(len\,(x))$ para algún polinomio P_L. Así, se llega a la siguiente formulación de un problema práctico:

Definición 3: Se denomina problema práctico (o simplemente problema) a la pareja $\langle R(x,y), P_L\rangle$, donde $R(x,y)$ es un algoritmo factible que transforma dos sucesiones binarias en un valor Booleano, verdadero o falso, y P_L es un polinomio.

Definición 4: Se denomina instancia de un problema $\langle R, P_L\rangle$ al siguiente problema:

"Dada una sucesión binaria x, generar o bien y tal que $R(x,y)$ es verdadero y $len(y) \leq P_L(len\,(x))$, o, si tal y no existe, un mensaje diciendo que no hay solución".

Por ejemplo, para el problema matemático descrito antes, una instancia puede ser: hallar una demostración de que x es verdadero o una refutación (una demostración de que $no\,x$ es verdadero).

2.3 LA CONJETURA DE P VS NP

En la literatura de optimización matemática o de complejidad computacional es común oír hablar de los "problemas prácticos generales", usualmente también descritos como "problemas de la clase NP", esta es una clasificación de los problemas de optimización para separarlos de aquellos problemas más complicados en los cuales la solución no es fácilmente verificable. Recordemos que los problemas tratables, o fáciles de resolver, son aquellos para los cuales existe un algoritmo factible que resuelve todas sus instancias.

La opinión generalizada es que no todos los problemas (prácticos generales) deben tener una solución fácil, de ahí que la conjetura que se maneja es que $NP \neq P$, pero nunca ha sido probada. Justamente esta conjetura es uno de los denominados "problemas del milenio" (1), establecidos por el Clay Mathematics Institute en el año 2000, y por cuya resolución se ofrece un premio de un millón de dólares por cada uno. Estos problemas del milenio son siete, y hasta el momento de escribir este artículo únicamente ha sido resuelta la denominada hipótesis de Poincaré, por el matemático ruso Grigori Perelman, logro que fue reconocido en el año 2010 (3) por lo que todavía seis de ellos permanecen abiertos, entre ellos la conjetura $NP \neq P$.

A modo de ejemplo, una manera de resolver un problema NP es chequear $R(x,y)$ para todas las sucesiones binarias y que satisfacen $len(y) \leq P_L\big(len(x)\big)$. Así, este algoritmo, denominado algoritmo del Museo Británico, requeriría un total de $2^{P_L(len(x))}$ chequeos, es decir es de tiempo exponencial, y por tanto no factible.

2.4 REDUCCIÓN DE UN PROBLEMA A OTRO

En términos informales, la manera como se demuestra que un problema pertenece a la clase NP es "*reduciéndolo*" a alguno de los que se conoce que pertenecen a esta clase. Una reducción aquí debe ser entendida como una transformación que no cambia la naturaleza de la complejidad del problema, y esto solo puede ser posible si se utiliza, de manera transitiva, una cadena de algoritmos factibles.

Como ejemplo, supongamos que se tiene un algoritmo que chequea cuando un sistema dado de desigualdades lineales es consistente. Y Consideremos adicionalmente otro problema, el de chequear cuando un sistema dado de desigualdades e igualdades es consistente. Es fácil concluir que éste último puede ser reducido al problema de la consistencia de un sistema de desigualdades, pues para esto basta reemplazar cada igualdad por dos desigualdades.

Definición 5: Dados dos problemas $\mathcal{P} = \langle R, P_L\rangle$ y $\mathcal{P}' = \langle R', P_L'\rangle$, se dice que $\mathcal{P}$ puede *reducirse* a $\mathcal{P}'$, si existen tres algoritmos factibles U_1, U_2 y U_3 con las siguientes propiedades:

- El algoritmo factible U_1 transforma cada entrada x del primer problema en una entrada del segundo problema.
- El algoritmo factible U_2 transforma cada solución y del primer problema en la solución del caso correspondiente del segundo problema; es decir, si $R(x,y)$ es verdadero, entonces $R'\big(U_1(x), U_2(y)\big)$ es también verdadero.
- El algoritmo factible U_3 transforma cada solución y' de la correspondiente instancia del segundo problema en la solución del primer problema; es

decir, si $R'(U_1(x), y')$ es verdadera, entonces $R(x, U_3(y'))$ también es verdadera.

Si existe una reducción de $\mathcal{P}$ a $\mathcal{P}'$, entonces una instancia x del primer problema es resoluble si y solo si la correspondiente instancia $U_1(x)$ del segundo problema es resoluble también. Por otra parte, si podemos resolver la segunda instancia (y encontrar una solución y'), entonces seremos capaces de encontrar una solución de la instancia original x del primer problema (como $U_3\ (y')$). Por lo tanto, si tenemos un algoritmo factible para resolver el segundo problema, sería posible diseñar un algoritmo para resolver el primer problema también.

fácilmente se puede ver como corolario que la relación de reducción satisface la propiedad transitiva; es decir, si un problema $\mathcal{P}$ puede reducirse a un problema $\mathcal{P}_0$, y el problema $\mathcal{P}_0$ se puede reducir a un problema $\mathcal{P}_1$, entonces, combinando estas dos reducciones, podemos concluir que $\mathcal{P}$ se puede reducir a $\mathcal{P}_1$.

A partir de este nuevo concepto se puede definir finalmente la NP-dureza de los problemas.

Definición 6: Un problema (no necesariamente de la clase NP) se denomina $NP-duro$ si cada problema de la clase NP puede ser reducida a él.

Definición 7: Si un problema de la clase NP es $NP-duro$, es llamado $NP-completo$.

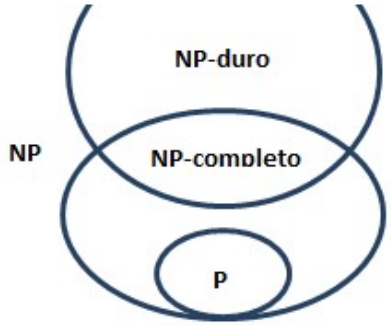
Según estas definiciones, si un problema $\mathcal{P}$ es $NP-duro$, entonces cada algoritmo factible para resolver este problema $\mathcal{P}$ daría lugar a algoritmos factibles para resolver todos los problemas de la clase NP; es decir, $NP = P$, lo cual se conjetura que es imposible. En vista de esta convicción, los problemas $NP-duros$ suelen ser denominados también intratables.

En resumen, aunque suele confundirse la terminología, los problemas de NP-completos son aquellos problemas NP-duros que están contenidos en NP, siendo los problemas NP-duros aquellos que tienen la propiedad que cualquier problema en NP puede ser transformado por medio de una reducción a uno NP-completo, es decir, si la conjetura $NP \neq P$ fuera cierta, los problemas estarían clasificados por su complejidad de acuerdo al gráfico, denominado diagrama de Euler, mostrado en la figura 1.

FIGURA 1

Algoritmos factibles, problemas tratables y la complejidad computacional de una variante del problema de la diversidad máxima

Diagrama de Euler con la clasificación de los problemas si $NP \neq P$



2.5 DEMOSTRACIÓN DEL CARÁCTER NP-DURO DE UN PROBLEMA

La demostración original del carácter NP-duro de cierto problema P_0 es bastante compleja, ya que se basa en demostrar explícitamente que todos los problemas de la clase NP se pueden reducir al problema P_0. Sin embargo, una vez que se ha demostrado la NP-dureza de un problema P_0, la demostración de que otro problema P_1es NP-duro es mucho más fácil a partir del concepto de reducción, ya que para demostrar que P_1es NP-duro, es suficiente demostrar que alguno de aquellos problemas que se conoce son NP-duros puede ser reducido a este problema P_1.

Históricamente, el primer problema que se demostró que es NP-completo fue el problema de satisfacibilidad booleana, lo que se conoce en la literatura como el teorema de Cook. Luego, en 1972, Richard Karp demostró que otros 21 problemas eran también NP-completos. Posteriormente se han descubierto que cientos de problemas pertenecen también a la clase de los NP-completo por reducciones desde otros problemas que previamente se habían demostrado NP-completos, muchos de estos problemas constan en el libro de Garey y Johnson (1). Entre los más conocidos de estos problemas están el problema de la mochila, el problema del agente viajero, el problema de la jorga máxima, etc. En particular, en este artículo, se utiliza el problema de la jorga máxima para demostrar que un nuevo problema combinatorio es NP-duro.

Teorema: Si un problema P_0 es NP-duro, cualquier problema más general P_1también es NP-duro.

Demostración: El teorema es demostrado en (1) de la siguiente manera:

El hecho de que P_0 es NP-duro implica que cada instancia p de cualquier problema P puede ser

reducida a alguna instancia p_0 del problema P_0. Ya que el problema P_1es más general que el problema P_0, cada instancia p_0 del problema P_0 es también una instancia del problema más general P_1. Así, cada instancia p de cualquier problema P puede ser reducida a alguna instancia p_0 del problema P_1, es decir el problema general P_1 es NP-duro.

El Problema de la Máxima Jorga: El problema de la máxima jorga es uno de los problemas que se conocen son NP-completos, y se define de la siguiente manera:

Sea un grafo G=(V,E), donde V es el conjunto de nodos y E es el conjunto de aristas, un subconjunto de nodos C⊆V se denomina una jorga de G si ∀ v_1,v_2∈C⊆V, existe una arista (v_1,v_2)∈E

La versión de decisión del problema de la máxima jorga consiste en chequear si:

"Dado un número k∈N ¿es posible encontrar una jorga C de tamaño al menos k en G?

3. EL PROBLEMA DEL MÁXIMO PROMEDIO Y SU CARÁCTER NP-DURO

Este problema del máximo promedio es un nuevo modelo del clásico problema de la diversidad máxima (4), que surge de la optimización de la medida de la diversidad promedio, y al contrario de otros modelos del problema de la diversidad máxima. En el problema del máximo promedio la cardinalidad del subconjunto seleccionado, M, es también una variable de decisión. Según puede observarse en (5) el problema del máximo promedio puede formularse como:

$$\max_{M\subset V,|M|\geq 2} \frac{\sum_{i<j,i,j\in M} d_{ij}}{|M|}$$

Donde V es un conjunto de elementos (población), donde se quiere seleccionar el subconjunto M más diverso.

Genéricamente hablando, este problema trata de maximizar la diversidad promedio. Una formulación de programación matemática con variables binarias es entonces la establecida por la formulación (3.1) –(3.3):

$$\max \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}x_i x_j}{\sum_{i=1}^{n} x_i} \qquad (3.1)$$

$$s.t. \quad \sum_{i=1}^{n} x_i \geq 2 \qquad (3.2)$$

$$x_i \in \{0,1\}, \quad 1 \leq i \leq n \qquad (3.3)$$

En este problema la función objetivo 0 es el promedio de la suma de las distancias entre los elementos seleccionados, la restricción 0 indica que por lo menos dos elementos deben seleccionarse. Tal como se presenta en (5), es un problema de optimización binaria fraccional

Propiedad :

Si los coeficientes d_{ij} no tienen restricciones en el signo, entonces el problema del máximo promedio es NP-duro.

Demostración:

Para la demostración consideramos como nuestro problema $\mathcal{P}_0$, al problema de la Máxima Jorga, el cual es conocido que es $NP-completo$, según puede consultarse en (1).

Sea $G=(V,E)$ un grafo, donde V es el conjunto de nodos y E es el conjunto de aristas, con $|V|=n$ nodos.

Consideremos ahora la siguiente instancia del problema del máximo promedio en su formulación con variables binarias:

$$\max_{x\in\{0,1\}^n,\ x\neq 0} g(x) = \frac{\sum_{i<j} d_{ij}x_i x_j}{\sum_{i=1}^{n} x_i}$$

Donde las distancias inter-elemento son tomadas como: $d_{ij} = \begin{cases} 1 & si\ (i,j)\in E \\ -n^2 & si\ (i,j)\notin E \end{cases}$

Entonces se tiene:

$$\max_{x\in\{0,1\}^n,\ x\neq 0} g(x) = \frac{-n^2\sum_{(i,j)\notin E, i<j} x_i x_j + \sum_{(i,j)\in E, i<j} x_i x_j}{\sum_{i=1}^{n} x_i}$$

Sea $x^*\in\{0,1\}^n, x^*\neq 0$. Definimos un subgrafo inducido $C(V_c,E_C)\subseteq G$, donde:

$$V_c = \{i\in\{1,\dots,n\}: x_i^* = 1\}$$

Si C no es una jorga: $x_i^*=1$, $x_j^*=1$ implica que $(i,j)\notin E$, entonces $g(x^*)<0$.

Por el contrario si C es una jorga: $\forall\,(i,j)$ tal que $x_i^*=1$, $x_j^*=1$ se tiene $(i,j)\in E$, y entonces:

$$g(x^*) = \frac{|C|(|C|-1)}{2|C|} = \frac{|C|-1}{2} \geq 0$$

Por lo tanto, el grafo G contiene una jorga de tamaño al menos k si y sólo si:

$$\max_{x\in\{0,1\}^n,\ x\neq 0} g(x) \geq \frac{k-1}{2}$$

Con lo que se demuestra que para esta instancia del problema de la diversidad máxima el problema se reduce a un problema de la máxima jorga, el cual se conoce, como se demuestra en (1), que es $NP-completo$.

REFERENCIAS BIBLIOGRÁFICAS Y ELECTRÓNICAS

[1]. GAREY, M., JOHNSON, D. (1979). "*Computers and Intractability: A Guide to the Theory of NP-Completeness*". s.l. : W.H. Freeman y Cía.

[2]. CLAY MATHEMATICS INSTITUTE. [En línea] [Citado el: 20 de 05 de 2013.] http://www.claymath.org/millennium/.

[3]. CARLSON, J. (2013). First Clay Mathematics Institute Millennium Prize Announced Today. [En línea] [Citado el: 22 de 05 de 2013.] http://www.claymath.org/poincare/millenniumPrizeFull.pdf.

[4]. GHOSH. (1996). "*Computational aspects of the maximum diversity problem*". 4, s.l.: Elsevier, Operations Research Letters, Vol. 19, págs. 175-181.

[5]. MARTÍ, R., SANDOYA, F. *GRASP and Path Relinking for the Equitable Dispersion Problem.* s.l. : Elsevier, 2012, Computers & Operations Research, págs. 1-18. http://dx.doi.org/10.1016/j.cor.2012.04.005.

[6]. PROKOPYEV, O., KONG, N., MARTÍNEZ-TORRES, D. *"The equitable dispersion problem*". 197, 2009, European Journal of Operational Research, págs. 59 - 67.