UNIVERSIDAD DE PANAMÁ
CENTRO REGIONAL UNIVERSITARIO DE VERAGUAS
FACULTAD DE INFORMÁTICA, ELECTRÓNICA Y COMUNICACIÓN

MONOGRAFÍA:

NP-COMPLETITUD: ORIGEN, FORMALIZACIÓN, CONSECUENCIAS Y ESTRATEGIAS DE SOLUCIÓN

PRESENTA:

RAÚL ENRIQUE DUTARI DUTARI

2006

TABLA DE CONTENIDOS

1. OBSERVACIONES PRELIMINARES.

Algoritmo es un conjunto finito de pasos que resuelve, sin ambigüedades, un problema dado.

La algoritmia es la ciencia que se encarga del diseño y el análisis de algoritmos específicos (cada vez más eficientes), que permiten resolver un problema pre-establecido.

Dentro de la algoritmia, se encuentra un tema especial: la complejidad computacional. Dentro de este tópico, se analizan, de manera general, todos los posibles algoritmos que son capaces de resolver un problema en particular, clasificándolos en función al tiempo de ejecución que requieren para resolver una instancia de un problema en particular (eficiencia en cuanto al tiempo de ejecución).

2. CLASIFICACIÓN DE LOS PROBLEMAS SEGÚN SU COMPLEJIDAD.

Según **SÁNCHEZ, J. (99),** Los problemas se pueden caracterizar, de acuerdo a su comportamiento para entradas de tamaño grande, como:

- **Problemas tratables:** Se caracterizan porque se conocen algoritmos razonablemente estudiados, que los resuelven de manera precisa y en un tiempo “razonable”.

- **Problemas tratables:** Se caracterizan porque se conocen algoritmos razonablemente estudiados, que los resuelven de manera precisa y en un tiempo “razonable”, con base en la cantidad de datos de entrada.

- **Problemas intratables:** Se conocen algoritmos que los pueden resolver de manera precisa; sin embargo, requieren una cantidad de tiempo de gigantescamente grande para resolver un problema, con datos de entrada relativamente pequeños.

- **Indecibibles o no computables:** Este tipo de problema se caracteriza porque, a la fecha, no se conocen algoritmos que permitan resolverlos; más aún, se desconoce si se puede construir un algoritmo que permita resolver dicho problema, de manera precisa.

Con respecto a los problemas no computables, la ciencia se encuentra en un estado de incomprensión significativo, prácticamente no se puede saber nada acerca de su solución. Uno de los problemas no computables más conocidos es el “Problema de la parada” de Alan Turing; así como el problema del castor laborioso, según **KORFHAGE (66)**.

En el otro extremo, los problemas tratables tienen un nivel de dificultad relativamente manejable, ya que el tiempo que necesita un computador para resolver un problema cualquiera de este tipo, crece dentro de límites razonables, a medida que aumenta la cantidad de datos de entrada que debe procesar. Son los clásicos problemas de ordenación por: inserción, burbuja, quicksort, por ejemplo.

Los problemas intratables son el eje de la exposición que se presenta a continuación. Primeramente, se considerará un concepto crucial en la temática: la NP-completitud.

3. ¿QUÉ ES LA NP-COMPLETIDUD?

La NP-Completidud es una propiedad no deseable de los algoritmos que "resuelven" algunos problemas de optimización y de toma de decisiones: los llamados problemas intratables.

Es un concepto utilizado en la Computación Teórica, en el Diseño y Análisis de Algoritmos. Fue descubierto por Stephen Cook, en 1970, durante sus estudios doctorales y representa un importante hito dentro de dicha rama de la Informática. La NP-Completitud es una aplicación del **Teorema de Cook**, que será mencionado oportunamente en esta monografía.

4. NP-COMPLETITUD: PUNTOS DE VISTA INFORMAL VS. FORMAL.

Los estudios formales acerca del problema de la NP-Completitud, normalmente se basan en la teoría de autómatas –Máquinas de Turing, particularmente– y requieren de una fuerte fundamentación matemática para comprenderlos a cabalidad.

Se puede visualizar una primera aproximación al concepto NP-Completidud, cuando **DEWDNEY, A.K. (89)** señala que:

> *"Si el algoritmo que se dispone para resolver un problema dado, requiere una cantidad extraordinariamente grande de tiempo, se considera como "Problema NP-Completo".*

Para facilitar la comprensión del tema al lector, esta monografía, presentará inicialmente un enfoque informal del problema, para posteriormente, enfocarlo formalmente.

4.1. ENFOQUE INFORMAL DE LA NP-COMPLETITUD.

Si consideramos que $n \in N$ es el tamaño de la entrada de un problema cualquiera. Los problemas que se resuelven en tiempo polinomial $O(n^k), \quad k \subset R$ se denominan de clase ***P***.

En tanto, aquellos problemas que se resuelven en un tiempo no polinomial – exponencial – $O(k^n), \quad k \subset R$, se denominan de clase ***NP***.

Para obtener una idea más clara del concepto, se presenta una tabla que contiene los tiempos de ejecución para una aplicación cualquiera. Cada columna representa los tiempos de ejecución de un algoritmo particular que se puede usar para programar una aplicación particular –de orden correspondiente al señalado en el encabezado–. En tanto que cada fila representa los tiempos de ejecución de todos los algoritmos, para una cantidad de datos definida.

Tiempo de ejecución Vs Tamaño de la entrada	**n**	**n^2**	**n^3**	**2^n**	**n!**	**n^n**
1	1	1	1	2	1	1
10	10	$1x10^2$	$1x10^3$	$1.02x10^3$	$3.63x10^6$	$1x10^{10}$
100	100	$1x10^4$	$1x10^6$	$1.26x10^{30}$	$9.33x10^{157}$	$1x10^{200}$
1000	1000	$1x10^6$	$1x10^9$	$1.07x10^{301}$	$4.02x10^{2567}$	$1x10^{3000}$

Para que el lector se pueda dar una perspectiva real de la magnitud de los valores que presenta la tabla anterior, **SÁNCHEZ, J. (99)** señala que el número de protones del universo es aproximadamente 10^{79} y el número de microsegundos desde el Big-Bang, es aproximadamente 10^{24}, ambas cifras, se observa, son enormemente menores que los tiempos de ejecución factoriales y exponenciales para entradas cercanas a 1000.

Sean segundos, o nano segundos los tiempos considerados, los algoritmos que presentan órdenes no polinomiales, se tornan inmanejables, a medida que crece el valor del n.

4.1.1. PROBLEMAS NP-COMPLETOS MÁS CONOCIDOS.

Los problemas NP-Completos se presentan en diversas disciplinas (más prácticas e importantes de lo que se desearía). Según **CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. (89),** los problemas NP-Completos se presentan al estudiar: Lógica Booleana, Grafos, Diseño de redes, planificación de horarios, optimización de programas, autómatas y lenguajes formales, entre otros dominios del conocimiento.

BAASE, S. (88), presenta algunos de los problemas NP-Completos más conocidos, que se enuncian a continuación.

4.1.1.1. EL PROBLEMA CNF-SATISFABILIDAD.

Históricamente fue el primer problema que se estudió (investigado por Cook en su Doctorado). Adicionalmente, desde el punto de vista teórico, es el más importante de todos los problemas NP-Completos, como será analizado posteriormente.

Este problema se describe con base a las definiciones que se enuncian a continuación:

- Un ***literal*** se define como una variable lógica o su negación.

- Una ***cláusula*** se define como una secuencia de literales separados por disyunciones (denotadas como $+$).

- Una ***expresión lógica*** se define que está en la ***forma normal conjuntiva (CNF)*** si es una secuencia de cláusulas separados por conjunciones (denotadas por $\bullet$).

- Si se considera que $a_j,\ \ 0 \le j \le n;\ \ 0 \le i \le m$ es una variable lógica, normal o negada, y se utilizan los símbolos convencionales de la matemática para representar las sumas y productos abreviados, entonces la expresión:

$$\prod_{i=0}^{m}\left(\sum_{j=0}^{n} a_j\right)_i$$

  Esta en ***forma normal conjuntiva***. De allí parte el planteamiento del problema CNF-satisfabilidad, como un problema de decisión:

Problema de Decisión: Se presenta una tabla de verdad arbitraria, formada variables lógicas o sus negaciones, asociadas por disyunciones y conjunciones. ¿Se pueden asignar valores arbitrarios a todas las variables lógicas involucradas en la tabla de verdad, de manera que siempre se reduzca a una tautología (o que siempre se exprese como una contradicción)?

Este problema encuentra aplicación en la prueba de teoremas computarizada, entre otros casos.

4.1.1.2. EL PROBLEMA DE LA COLORACIÓN DE GRAFOS.

El problema de la coloración de grafos, se puede plantear de dos formas: como un problema de decisión (similar al CNF-satisfabilidad), o como un problema de optimización. A continuación, se presentan ambos planteamientos:

Problema de decisión: Dado el grafo G y una cantidad k de colores: ¿Se puede pintar G de modo no se presenten regiones adyacentes con el mismo color?

Problema de optimización: Dado un grafo $G = \{a_1, a_2, ..., a_n, \}$ determine la menor cantidad de colores k que se necesita para pintar a G; de manera que: no se presenten dos regiones adyacentes $a_i, a_j,$ con $i \neq j$, tales que $C(a_i) \neq C(a_j)$ – es decir, que las regiones adyacentes se pinten utilizando el mismo color –.

El problema de coloración de grafos es una abstracción de ciertos tipos de problemas de planificación. Se presenta en la confección de horarios, entre otros casos (al evitar el choque de fechas o aulas, o al confeccionar horarios de clase, por ejemplo).

4.1.1.3. EL PROBLEMA DEL EMBALAJE DE CAJAS.

Se suponen un número ilimitado de cajas con capacidad 1, y n objetos de tamaños $s_1, s_2, ..., s_n$, donde $0 < s_j \leq 1$.

Problema de Optimización: Se debe determinar cual es la menor cantidad de cajas k que se requiere para empacar los n objetos.

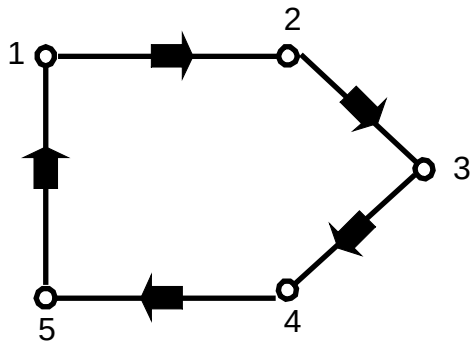
Problema de Decisión: Dados n objetos de tamaños $s_1, s_2, ..., s_n$, donde $0 < s_j \leq 1$ y k cajas, se debe determinar si los n objetos se pueden empacar dentro de las k cajas.

Las aplicaciones de este problema incluyen: Empaquetamiento de datos en memoria de computadoras. Despacho de órdenes de un producto (tal como tejido o madera), que se produce con una medida estándar.

4.1.1.4. EL PROBLEMA DEL CIRCUITO HAMILTONIANO.

Un circuito hamiltoniano en un grafo o dígrafo; es un recorrido que pasa exactamente una vez por cada vértice.

Ejemplo: la figura que presentamos a continuación muestra un circuito hamiltoniano.



Problema de Decisión: Dado un grafo o dígrafo cualquiera: ¿Posee un circuito hamiltoniano.

4.1.1.5. EL PROBLEMA DEL AGENTE VIAJERO.

Un problema relacionado al de los circuitos hamiltonianos, es el denominado “Problema del Agente Viajero”.

Problema de Optimización: Dado un grafo G, con pesos enteros en sus lados, se debe encontrar un ciclo que incluya todos los nodos de G tal que la suma de todos los pesos del ciclo sea menor o igual que un k dado.

4.2. DESARROLLO FORMAL DEL CONCEPTO NP-COMPLETITUD.

Para lograr la compresión cabal de la teoría relacionada con la NP-completitud, se deben realizar algunas consideraciones teóricas importantes. Este desarrollo teórico está fundamentado en **KELLEY, D. (95).**

Dado que esta es una teoría bastante extensa y compleja, solamente se expondrán sus resultados más relevantes sin las correspondientes demostraciones. El lector interesado en sus detalles puede consultar las referencias bibliográficas, en particular **KELLEY, D. (95).**

4.2.1. REDUCIBILIDAD EN TIEMPO POLINÓMICO.

Definición (1): f es una función de cadena computable en tiempo polinomico, si existe una máquina de Turing M con cota temporal polinómica, tal que, dada una cadena de entrada w, calcula u, siempre que $f(w) = u$.

Definición (2): Un lenguaje L_1 es reducible en tiempo polinómico a un lenguaje L_2 si hay una función de cadena computable en tiempo polinómico f, tal que $f(u) \in L_2$ si y sólo si $u \in L_1$.

4.2.2. LAS CLASES P Y NP.

Definición (3): La clase P se compone de todos los lenguajes que aceptan una máquina de Turing determinista que tiene una cota temporal polinómica.

Definición (4): La clase NP se compone de todos los lenguajes que aceptan una máquina de Turing determinista que tiene una cota temporal exponencial (o no determinista en tiempo polinomial).

4.2.3. TEOREMA DE LA REDUCIBILIDAD.

Sea L_1 un lenguaje reducible en tiempo polinómico a otro lenguaje L_2. Entonces:

- Si $L_2 \in P$, entonces $L_1 \in P$.
- Si $L_2 \in NP$, entonces $L_1 \in NP$.

Notación: Si L_1 es reducible en tiempo polinómico a L_2, se denotará como: $L_1 <_P L_2$.

4.2.4. LA CLASE C-DIFÍCIL.

Definición (5): Para cualquier clase de lenguajes C, un lenguaje L se dirá *C*-difícil si para todo $L' \in C, \quad L' <_P L$. En particular, L es NP-difícil si para todo lenguaje $L' \in NP, \quad L' <_P L$.

Observación: la definición no establece donde está L; puede o no estar en C.

4.2.5. LA CLASE NP-COMPLETA.

Definición (6): Si L es *C*-difícil y $L \in C$, entonces L es *C*-completo.

Definición (7): En particular, si L es *NP*-difícil y $L \in NP$, entonces L es *NP*-completo.

4.2.6. EL PROBLEMA DE LA IGUALDAD ENTRE P Y NP.

Teorema (2): Si L es un lenguaje *NP*-completo y $L \in P$, entonces $P \subseteq NP$.

Una de las más importantes incógnitas dentro de esta teoría, es lo referente al problema: ¿P=NP?. Se sospecha que NP es un conjunto bastante más grande que P; y adicionalmente, muchos expertos matemáticos e informáticos han intentado sin éxito, la comprobación de la inclusión $NP \subseteq P$, por lo que el sentido común sugiere que esta inclusión no se puede comprobar. Sin embargo, tampoco se ha podido refutar, de modo que es un problema abierto a la búsqueda de soluciones -según **BAASE, S.(88)-.**

En otro sentido, la pregunta: ¿Qué lenguaje es NP-completo?, puede ser respondida satisfactoriamente, como será expuesto a continuación.

4.2.7. RELACIÓN ENTRE PROBLEMAS Y LENGUAJES NP-COMPLETOS.

El problema CNF-satisfabilidad, puede ser codificado, de modo que genere el lenguaje NP deseado. En tal sentido, se debe recordar la expresión de dicho problema:

$$\prod_{i=0}^{m}\left(\sum_{j=0}^{n} a_j\right)_i$$

4.2.7.1. CODIFICACIÓN DEL PROBLEMA CNF-SATISFABILIDAD.

a : se codifica como *"&"*

i, j j: se codifican en binario como cadenas de 0's y 1's.

El valor de la expresión también puede ser binario.

El alfabeto será entonces:

$$\Sigma = \{",",")","+",")","\cdot",")","\&",")","0",")","1"\}$$

El lenguaje así definido se denotará como:

$$L_{sat} = \{w \in \Sigma^* / \, w \text{ es un conjunto satisfactible de cláusulas}\}$$

Bajo la construcción anterior se puede probar que: $L_{sat} \in NP$ y que:

L_{sat} es NP-completo. Este último resultado es conocido como el **Teorema de Cook.**

4.2.7.2. REDUCCIÓN DE LENGUAJES.

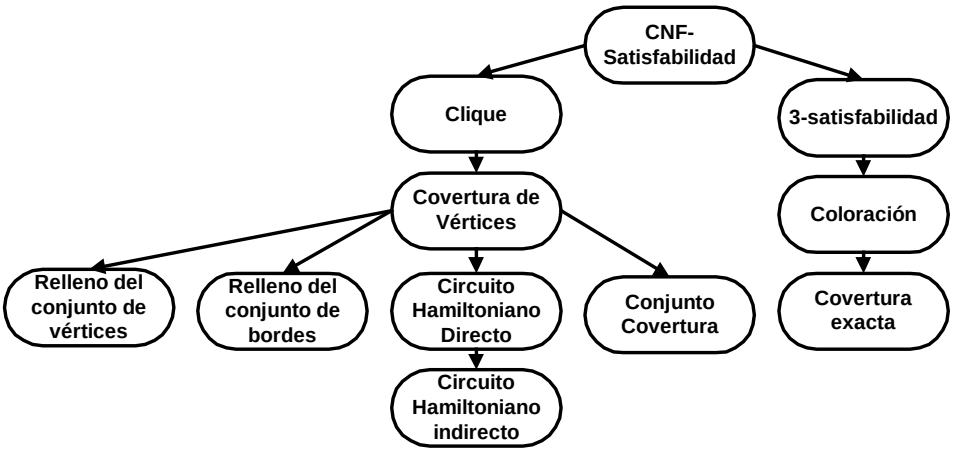
La reducción de lenguajes proporciona la herramienta necesaria para demostrar que otros lenguajes son NP-completos. De allí se desprende:

Lema: Si L_1 es NP-completo y $L_1 <_p L_2$, entonces L_2 es NP-difícil.

Corolario: Si L_1 es NP-completo y $L_2 \in NP$ con $L_1 <_p L_2$, entonces L_2 es NP-completo.

5. OTROS PROBLEMAS NP-COMPLETOS.

Vía la reducción de lenguajes, los estudiosos del tema de la NP-completitud han podido recopilar una lista de alrededor de 2000 problemas del tipo NP-Completo. Todos han sido reducidos al problema de CNF-Satisfabilidad. Entre ellos, los más relevantes se presentan en el siguiente diagrama:



6. CONSECUENCIAS DE LA EXISTENCIA DE LOS PROBLEMAS NP-COMPLETOS.

Según **LEVINE, G. (97)**: los problemas NP-Completos representan una barrera en la solución más general de ciertos tipos de problemas.

Aún, si las computadoras del futuro fueran cientos de millones de veces más rápidas que las que tenemos actualmente, no podrían resolver estos problemas, empleando un tiempo razonablemente corto.

En pocas palabras, representan en la actualidad, lo que los problemas clásicos representaron para los griegos[1], un límite a el crecimiento de la ciencia.

7. ESTRATEGIAS PARA "RESOLVER" LOS PROBLEMAS NP-COMPLETOS.

Según **BAASE, S. (88),** existen cientos de problemas de aplicación importantes que son NP-completos. Para entradas razonablemente grandes, las soluciones exactas son imposibles de obtener.

Bastará con lograr resolver uno de los problemas en el caso general, para que a través de las reducciones polinomiales se puedan resolver los restantes.

Las soluciones aproximadas, generalmente, se enfocan en el desarrollo e investigación de tópicos tales como:

---

[1] Cuadratura del círculo, trisección del ángulo, la duplicación del cubo.

- Algoritmos aproximados.
- Heurísticas.
- Algoritmos probabilísticos.
- Programación paralela.

Este es un campo de investigación abierto en la actualidad.

8. CONCLUSIONES.

- La NP-Completitud es una propiedad no deseable que presentan los algoritmos que conocemos para resolver ciertos tipos de problemas.
- La existencia de problemas con estas características es un hecho que se presenta intelectualmente amenazador.
- Existen tópicos en esta temática, donde la investigación científica está en pleno desarrollo ($P = NP$).
- La reducibilidad de lenguajes garantiza que la solución completa de uno sólo de los problemas NP-Completos, puede ser extrapolada exitosamente para resolver los restantes problemas NP-Completos.
- El Teorema de Cook garantiza la existencia de, al menos, un lenguaje NP-Completo.
- Existen métodos que permiten aproximar soluciones para ciertas instancias de algunos problemas NP-completos.

9. REFERENCIAS BIBLIOGRÁFICAS.

1. **AHO, A. V.; HOPCROFT, J. E.; y ULLMAN, J. D.** *Estructuras de Datos y Algoritmos.* México. Addison-Wesley. 1988.

2. **BAASE, S.** *Computer Algorithms: Introduction to Desing and Analys.* Second Edition. U.S.A. Addison-Wesley, 1988.

3. **BRASSARD, G. Y BRATLEY; P.** *Fundamentos de Algoritmia.* España, Prentice-Hall, 1997.

4. **CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L.** *Introduction to algorithms.* U.S.A. MIT Press. 1989.

5. **DEWDNEY, A.K.** *The Turing Omnibus: 61 excursions in Computer Science.* U.S.A. Computer Sciencie Press, 1989.

6. **HOPCROFT, J. E.; MOTWANI, R. y ULLMAN, J. D.** *Introducción a la Teoría de Autómatas, Lenguajes y Computación.* Segunda Edición. España. Addison-Wesley. 2001.

7. **HORWITZ, E. & SAHNI, S.** *Fundamentals of computer algorithms.* U.S.A. Computer Sciencie Press, 1978.

8. **KELLEY, D.** *Teoría de Autómatas y Lenguajes Formales.* España. Prentice Hall, 1995.

9. **KORFHAGE, R.R.** *Lógica y Algoritmos con aplicaciones a las ciencias de la computación e información.* México. Limusa-Wiley, 1966.

10. **LEVINE, G.** *Estructuras fundamentales de la computación: Los principios.* México. McGraw-Hill, 1997.

11. **SÁNCHEZ, J.** *Curso de Análisis de Algoritmos.* México. Instituto Tecnológico y de Estudios Superiores de Monterrey, 1999.