Asignatura: Estructura de Datos y Algoritmia
Profesor: D. Sc. Gerardo García Gil
2021-B
Alumno: José Rafael Ruiz Gudiño
Ingeniería en Desarrollo de Software
Centro de Enseñanza Técnica Industrial (CETI)

1.- Problema de satisfacibilidad booleana (SAT)

En teoría de la complejidad computacional, el Problema de satisfacibilidad booleana (también llamado SAT) fue el primer problema identificado como perteneciente a la clase de complejidad NP-completo.

Su NP-completitud fue demostrada por Stephen Cook en 1971 (el Teorema de Cook).1 Hasta entonces el concepto de problema NP-completo no había sido definido. El SAT sigue siendo NP-completo incluso si todas las fórmulas están en forma normal conjuntiva (FNC) con 3 variables por cláusula (3SAT-FNC) creando el problema (3SAT), o aun en el caso de que solo se permita un único valor verdadero en cada cláusula (3SAT en 1).

Comenzamos con una lista de variables booleanas x1, …, xn. Un literal es una de las variables xi (o la negación de una de las variables ¬xi). Hay 2n literales posibles. Una cláusula es un conjunto de literales.

Las reglas son las siguientes: Se asignan valores booleanos Verdadero (V) o Falso (F) a cada una de las variables. De este modo a cada uno de los literales se le asigna un valor booleano. Finalmente, una cláusula tiene valor V si y sólo si al menos uno de los literales de la cláusula tiene un valor V, en otro caso tendrá un valor F.

Un conjunto de cláusulas es satisfactible si existe una asignación de valores booleanos a las variables que hagan que todas las cláusulas sean ciertas. Consideramos or (v) entre cada uno de los literales en una cláusula y and (^) entre las cláusulas. El problema de satisfacibilidad (SAT). Dado un conjunto de cláusulas. ¿Existe un conjunto de valores booleanos para una determinada expresión que la haga verdadera?

Ejemplo: Consideramos el conjunto de variables x1, x2, x3. Podemos construir la siguiente lista de cláusulas.

{x1, ¬x2} {x1, x3} {x2, ¬x3} {¬x1, x3}

Si elegimos los valores (V,V,F) para las variables (x1, x2, 3) respectivamente, entonces los valores de las cuatro cláusulas será (T,T,T,F), así que no podría ser una asignación válida para satisfacer el conjunto de cláusulas. Existen 8 posibles asignaciones (2n=3). Al final obtenemos como asignación satisfactoria a (T,T,T).

El ejemplo nos deja la sensación de que SAT debe ser un complicado problema computacional, porque hay 2^n posibles conjuntos de valores que pueden resolver el problema. Está absolutamente claro, sin embargo, el problema pertenece a la clase de complejidad NP. Efectivamente, es un problema de decisión. Además, podemos asignar fácilmente un certificado a todos los conjuntos de cláusulas para cual la respuesta a SAT es 'Sí, las cláusulas son satisfactibles'. El certificado contiene un conjunto de valores, uno por cada variable, que satisface todas las cláusulas. Este problema tiene una notación asintótica $O(n^k)$, es decir, de complejidad polinomial la cual mientras mas incremente k el algoritmo se volverá más lento.

En 3-SAT, transiciones rápidas en la solubilidad de una instancia puede observarse. Este fenómeno es llamado transición de fase (del inglés phase transition). Si la instancia tiene pocas cláusulas se dice que es no-restringida y es fácil encontrar un modelo. Si hay un gran número de cláusulas se dice que la instancia es sobre-restringida, y una buena heurística será capaz de recortar rápidamente gran parte del árbol de búsqueda. La transición de fase ocurre cuando se pasa de instancias no-restringidas a instancias sobre-restringidas (CHEESEMAN, et al., 1991). Las instancias más difíciles se encuentran entre estos dos tipos de instancias, donde el 50\% de ellas es satisfacible con aproximadamente m/n » 4.3. Este valor se conoce como punto de cruce.

2.- Problemática de P, NP y NP Completos

P

Gran parte de la ciudadanía tiende a pensar que los computadores pueden resolver todos los problemas, y que los que no pueden resolver hoy, podrán hacerlo mañana, porque su potencia de cálculo crece continuamente. En cambio, una infinidad de problemas de cómputo no tendrán solución nunca (los llamamos problemas indecidibles), y que para otros problemas existen algoritmos que los resuelven, pero empleando para ello tanto tiempo de cálculo, que a efectos prácticos es como si fueran irresolubles (los llamamos problemas intratables). Los problemas que resuelven los computadores en un tiempo razonable los llamamos polinomiales, y todos ellos se agrupan en la llamada clase P. Se dicen así porque su tiempo de cómputo está descrito por un polinomio en el tamaño de los datos. Por ejemplo, el problema de multiplicar dos matrices de n filas y n columnas se puede resolver utilizando menos de n3 multiplicaciones. Ninguno de los problemas intratables está en la clase P.

NP

Hay otra clase de problemas, a la que llamamos NP, cuya definición está hecha de tal manera que incluye todos los problemas de la clase P, pero también otros muchos que se comportan de un modo intrigante. Uno de esos problemas es el llamado problema del viajante de comercio: dado un mapa de carreteras, consiste en encontrar el camino más corto para visitar n ciudades una sola vez y volver al punto de origen. Para estos nuevos problemas de la clase NP, los mejores algoritmos que se conocen tienen un coste similar al de los problemas intratables, pero nadie ha podido demostrar que no existan algoritmos polinomiales para ellos. Tampoco nadie ha demostrado que sean intratables. Están, por decirlo así, en una especie de limbo informático: no se sabe si son polinomiales, o si son intratables.

NP-Completos

La teoría desarrollada en estos años ha llegado sin embargo a alguna conclusión útil: ha definido una subclase de la clase NP, la subclase de los problemas NP-completos, en la cual se agrupan los problemas más costosos de la clase NP, de tal forma que, si para uno cualquiera de dichos problemas se encontrara un algoritmo polinomial, entonces todos ellos se resolverían en tiempo polinomial y además la clase NP colapsaría a P, es decir tendríamos la igualdad P=NP. Más aún, si se demostrara que uno solo de los problemas NP-completos es intratable, entonces todos ellos lo serían y tendríamos la desigualdad P≠NP.

Problemática

Las consecuencias de esto último no serían muchas: simplemente dejaríamos de buscar algoritmos polinomiales para una serie de problemas interesantes, porque sabríamos con seguridad que tales algoritmos no existen. En cambio, si fuera P=NP, habríamos encontrado algoritmos polinomiales para todos esos problemas. La parte buena de ello es que podríamos resolver, en tiempos muy cortos, problemas del viajante con miles de ciudades y otros cientos de problemas útiles para los que hoy tenemos algoritmos muy costosos, y eso sería beneficioso para la industria, las comunicaciones y el desarrollo en general. La parte mala es que las claves criptográficas se descifrarían con gran facilidad, y muchas cuentas bancarias y comunicaciones cifradas quedarían expuestas a los amigos de lo ajeno.

En efecto, la criptografía actual depende de un problema de la clase NP, el de la descomposición en factores de un número, para el que no tenemos algoritmos eficientes. El más eficiente de todos tardó 18 meses en descomponer en factores un número de 200 cifras decimales, que son los que se usan habitualmente en criptografía. La seguridad de las claves descansa precisamente en esta dificultad, hoy por hoy insalvable. Si fuera P=NP, entonces la descomposición en factores pasaría a ser un problema polinomial y se podría resolver eficientemente. La criptografía tendría que ingeniárselas para basar la seguridad de sus claves en la resolución de algún problema realmente intratable, porque los de la clase NP habrían pasado todos ellos a la categoría de eficientes.

Referencias

Jiménez, A. (2006, 10 octubre). *Problema de satisfacibilidad (SAT)*. Xataka Ciencia. https://www.xatakaciencia.com/computabilidad/problema-de-satisfacibilidad-sat

Peña Marí, R. (2017, 22 mayo). *El problema que los informáticos no han podido resolver en 45 años*. EL País. https://elpais.com/tecnologia/2017/05/19/actualidad/1495202801_698394.html

Villagra, M., & Barán, B. (s. f.). *Análisis de Satisfacibilidad en Lógica Proposicional*. Universidad Nacional de Asunción. Recuperado 4 de septiembre de 2021, de http://sdi.cnc.una.py/catbib/documentos/000311.pdf