Introducción a las técnicas heurísticas

Semana 2

Dr. Roman Anselmo Mora Gutierréz

8 de enero de 2020

Universidad Autónoma Metropolitana

Tabla de contenido

1. Problemas P, NP y NP-completos

2. Búsqueda Local

Problemas P, NP y NP-completos

Problemas de decisión

Un problema de decisión es aquel cuya solución es simplemente "sí" o "no" [1, 2]. Formalmente, un problema de decisión Π consiste simplemente en el conjunto de D_Π formado por las instancias y un subconjunto de las $Y_\Pi \subseteq D_\Pi$ de instancias cuya respuesta es afirmativa [1].

Problemas de decisión

A continuación se expone un ejemplo de un problema de decisión

Ejemplo

Problema de decisión del agente viajero. Instancias: Un conjunto finito de ciudades $C = \{c_1, c_2, c_3, \ldots, c_m\}$, *una función de distancia* $d(c_i, c_j) \in Z^+$ *para cada par de ciudades* $c_i, c_j \in C$ *y un valor* $B \in Z^+$. *Pregunta: Existe un tour entre todas las ciudades en* C, *tal que la longitud total de éste no sea mayor a* B; *es decir, existe un ordenamiento de la ciudades en* $C < c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(m)} >$ *tal que* $[\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)})] + d(c_{\pi(m)}, c_{\pi(1)}) \leq B$.

En el ejemplo 1, el problema de decisión es derivado del correspondiente problema de optimización, (agente viajero _TSP por la siglas en inglés de *travelling salesman problem* _); el cual requiere encontrar el tour con costo mínimo, puede ser asociado con un problema de decisión; donde se incluya una condición de forzamiento tal que el conjunto de recorridos posibles quede restringido a aquellos cuyo valor sea menor o igual a B.

Problemas de decisión

Existe una correspondencia entre los problemas de optimización y los problemas de decisión; es decir, un problema de optimización Π se expresa como un problema de decisión Π' con la imposición de un valor límite sobre la función objetivo [3].

Generalmente, los problemas de optimización son más difíciles de resolver que sus equivalentes problemas de decisión [2]; ya que en el problema de optimización se necesita para resolverlo como mínimo la misma cantidad de recursos utilizados para solucionar el correspondiente problema de decisión.

Esta relación resulta sumamente importante ya que la teoría sobre las clases de problemas P, NP y NP-completos ha sido desarrollada a partir de los problemas de decisión. A continuación, se abordaran una serie de conceptos necesarios para abordar la complejidad de los problemas

Máquina de Turing y lenguaje

Generalmente, se recurre a los conceptos de lenguaje y máquina de Turing, para describir y analizar a los problemas decisión, puesto que cualquiera de estos problemas se puede representar como "lenguaje", siendo equivalente el proceso de resolver el problema con el proceso de reconocimiento del correspondiente lenguaje [1].

Máquina de Turing y lenguaje

Definición

Sea $\sum$ *un alfabeto* [1] *con al menos dos símbolos y sea* $\sum*$ *el conjunto finito formado por todas las cadenas finitas generadas a partir de* $\sum \bigcup \varnothing$. *Un lenguaje L sobre* $\sum$ *es un subconjunto* $\sum*$ *[1, 4].*

[1] Un alfabeto es un conjunto finito de símbolos, diferente al conjunto vacío.

Máquina de Turing y lenguaje

Ejemplo

Dados $\sum = \{0, 1\}$ *entonces* $\sum *$ *contendrá a las cadenas* $\varnothing, 0, 1, 00, 01, 10, 11, 000, 010, 001, \ldots$, *si se define a L como un subconjunto de* $\sum *$ *tal que L contenga a todas las cadenas de no más de dos elementos entonces* $L = \{\varnothing, 0, 1, 00, 01, 10, 11\}$.

Máquina de Turing y lenguaje

La correspondencia entre los problemas de decisión y el lenguaje está dada por el esquema de codificación especificado en el problema; el esquema de codificación e de un problema D_Π provee la manera apropiada para describir a cada una de las instancias de D_Π en cadenas de símbolos sobre un Σ; por lo tanto, la codificación genera una partición en tres clases del conjunto $\Sigma\star$, que son: A) las cadenas no codificadas para ninguna instancia de D_Π, B) aquellas codificaciones de las instancias de D_Π cuya respuesta es "no" y C) aquellas codificaciones de las instancias de D_Π cuya respuesta es "sí". Esta tercera clase de cadenas es el lenguaje asociado a D_Π y e.

Máquina de Turing y lenguaje

$$L[D_\Pi, e] = \left\{ \; x \in \Sigma\star \quad \begin{array}{c} \Sigma \text{ sea el alfabeto usado por} \\ e \text{ y } x \text{ sea una codificación dentro de } e \\ \text{para una instancia } I \text{ tal que } I \in Y_\Pi \end{array} \right\} \quad (1)$$

Máquina de Turing y lenguaje

Una máquina de Turing M [2] es un dispositivo teórico computacional, compuesta por una unidad de control de estados finitos (ejemplo algoritmo) y una cinta [3] magnética en ambas direcciones infinitas, la comunicación entre la unidad de control y la cinta sólo se da a través de una cabeza de lectura-escritura, la cual, es capaz de moverse en ambas direcciones de la cinta a una celda adyacente, posteriormente el cabezal puede leer y escribir los símbolos de un alfabeto. Una máquina de Turing realiza una tarea a través de combinar algunas de las siguientes operaciones: a) lee un caracter en la cinta, b) efectúa una transición de estado, c) realiza una acción en la cinta.

[2] Concepto introducido por Allan Turing en 1937

[3] La cinta está dividida en celdas, cada una de ellas es capaz de almacenar un símbolo de una alfabeto prefijado $\sum \bigcup \varnothing$.

Máquina de Turing y lenguaje

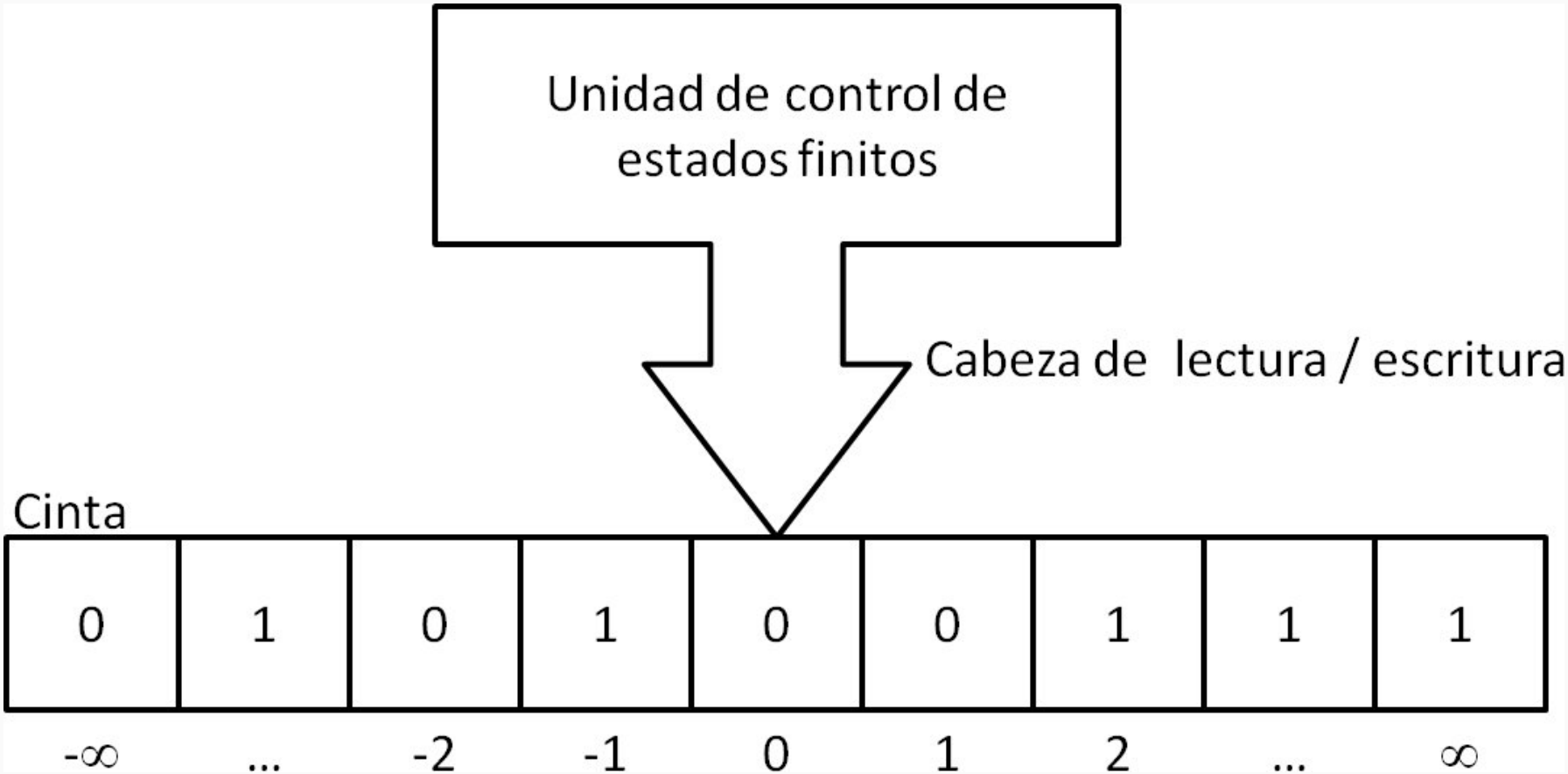

Figura 1: Visualización de una máquina de Turing.

Máquina de Turing y lenguaje

Definición

Una MT es un quíntuplo $(K, \Sigma, \Gamma, Q, \delta)$ *donde:*

- K *es un conjunto de estados tal que* $h \in K$.
- Σ *es el alfabeto de entrada, donde el caracter en blanco* $\sqcup \notin \Sigma$.
- Γ *es el alfabeto de la cinta.*
- δ *es el estado inicial.*
- Q *es la función de transición.*

Máquina de Turing y lenguaje

La operación que debe realizar M está determinada por su función de transformación; dado un estado y valor actual se produce un nuevo estado y valor a través de su movimiento en una dirección. Es decir, dada la función de transición $Q(q, s) = (q', s', l)$, ésta indica que M está actualmente en el estado q escaneando el símbolo s; en el nuevo estado q', el cabezal de la cinta se habrá movido hacia la derecha o hacia la izquierda dependiendo si l es -1 o 1, colocándose en una nueva celda donde se imprimirá el símbolo s'.

Máquina de Turing y lenguaje

Se dice que M corre en un tiempo polinomial si existe una constante k tal que para toda n se satisfaga que $T_M(n) \leq n^k + k$ [4]. Si para cada $\delta(q, s)$ sólo existe una posibilidad de ejecución, se trata de una máquina de Turing determinista, mientras que en el caso de que dado un $\delta(q, s)$ existan dos o más posible acciones se trata de una máquina de Turing no determinista; a continuación se formalizan estas definiciones:

Máquina de Turing y lenguaje

Definición

Dada una máquina de Turing si la función de transformación se define como $\delta(q, s) = (q', s', h)$ *para todo estado* q *y para cada símbolo en la cinta* s*, entonces* M *es una máquina de Turing determinista (MD).*

Definición

Dada una máquina de Turing si la función de transformación se define como
$\delta(q, s) = \{(q'_1, s'_1, h_1), (q'_2, s'_2, h_2), \ldots, q'_k, s'_k, h_k\}$ *donde:*$k \in \mathbb{N}, k \geqq 2$*,para todo estado* q *y para cada símbolo en la cinta* s *entonces* M *es una máquina de Turing no determinista (MND).*

Tesis de Turing

Definición

La **Tesis de Turing** *establece que las funciones que pueden ser calculadas mediante un método definido coincide con la clase de las funciones calculables mediante una Máquina de Turing.*

Corolario

Un problema tiene solución si existe una máquina de Turing capaz de calcularla [5].

Tesis de Church

Definición

Cuando existe un método efectivo (algoritmo) para obtener valores de una función matemática, la función puede ser calculada por una MT [5]. En otras palabras, toda función computable es recursiva y todo conjunto decidible es recursivo [6].

Las ideas propuestas por Turing y Church fincaron el **paradigma de computación clásica**.

Problemas indecidible

Son aquellos problemas que no pueden solucionarse en forma algorítmica; generalmente a estos problemas se les puede describir, pero no se pueden representar o resolver. Algunos ejemplos de problemas indecidibles son: A) Dada una M arrancada, determinar si M se detiene con la cinta vacía; B) dada una M, determinar si M se detiene frente alguna (o todas) las entradas posibles; C) dada una M, determinar si el lenguaje que acepta M es finito; D) dadas M_1 y M_2, determinar si ambas máquinas aceptan el mismo lenguaje; E) dadas M_1 y M_2, determinar si se detienen frente la misma entrada [7].

Problemas Decidibles

Son aquellos problemas para los cuales existe al menos un algoritmo que puede decidir para cada posible frase de entrada si dicha frase pertenece o no al lenguaje del problema de decisión.

Definición
Un **problema de decisión** Π *es un par* (L_X, L_Y)*; donde:* L_X *es un lenguaje decidible en tiempo polinomial y* $L_Y \subseteq L_X$ *. Los elementos de* L_X *son llamados instancias de* Π*; los elementos de* L_Y *son las instancias afirmativas de* Π *-denotadas como sí-instancias- ; mientras que* $L_X \setminus L_Y$ *son las instancias negativas de* Π *-denotadas como no-instancias- . Un* **algoritmo para un problema de decisión** (L_X, L_Y) *es un algoritmo capaz de calcular una función* $f : L_X \to \{0, 1\}$ *tal que:* $f(x) = 1$ *si* $x \in L_Y$ *y* $f(x) = 0$ *si* $x \in L_Y \setminus LX$ *[8].*

Problemas Decidibles

Clase P

Son aquellos problemas de decisión que pueden solucionarse por un algoritmo con un número de pasos limitados por un polinomio en función del tamaño de entrada, en términos de lenguaje la clase P se define como:

$$P = \{L | L = L(M) \text{ para alguna máquina M de Turing que corre en tiempo polinomial}\} \tag{2}$$

Problemas Decidibles

Clase NP

Es la colección de aquellos problemas de decisión, para los cuales, sólo se conocen algoritmos de solución no-determinísticos en tiempo polinomial; lo anterior, indica que una M no determinística resuelve en tiempo polinomial esta clase de problemas.
Otra definición equivalente utiliza el lenguaje L_R de una relación binaria R de chequeo entre dos lenguajes[4], el lenguaje L_R se define como:
$L_R = \{w\sharp y | R(w, y)\}$

[4]Dado $R \subseteq \Sigma * \times \Sigma *_1$, sea L_R el lenguaje definido $\Sigma \bigcup \Sigma_1 \bigcup \{\sharp\}$ donde:$\sharp$ indica que el símbolo no esta Σ

Relación entre clase P y NP

Existe una relación entre las clases P y NP; ya que, dado un problema $\Pi \in P$ dicho problema Π pertenecerá también a NP. En otras palabras, si Π es un problema de decisión que se soluciona a través de un algoritmo determinístico en tiempo polinomial, entonces Π puede ser resuelto en un tiempo polinomial por un algoritmo no determinístico

Relación entre clase P y NP

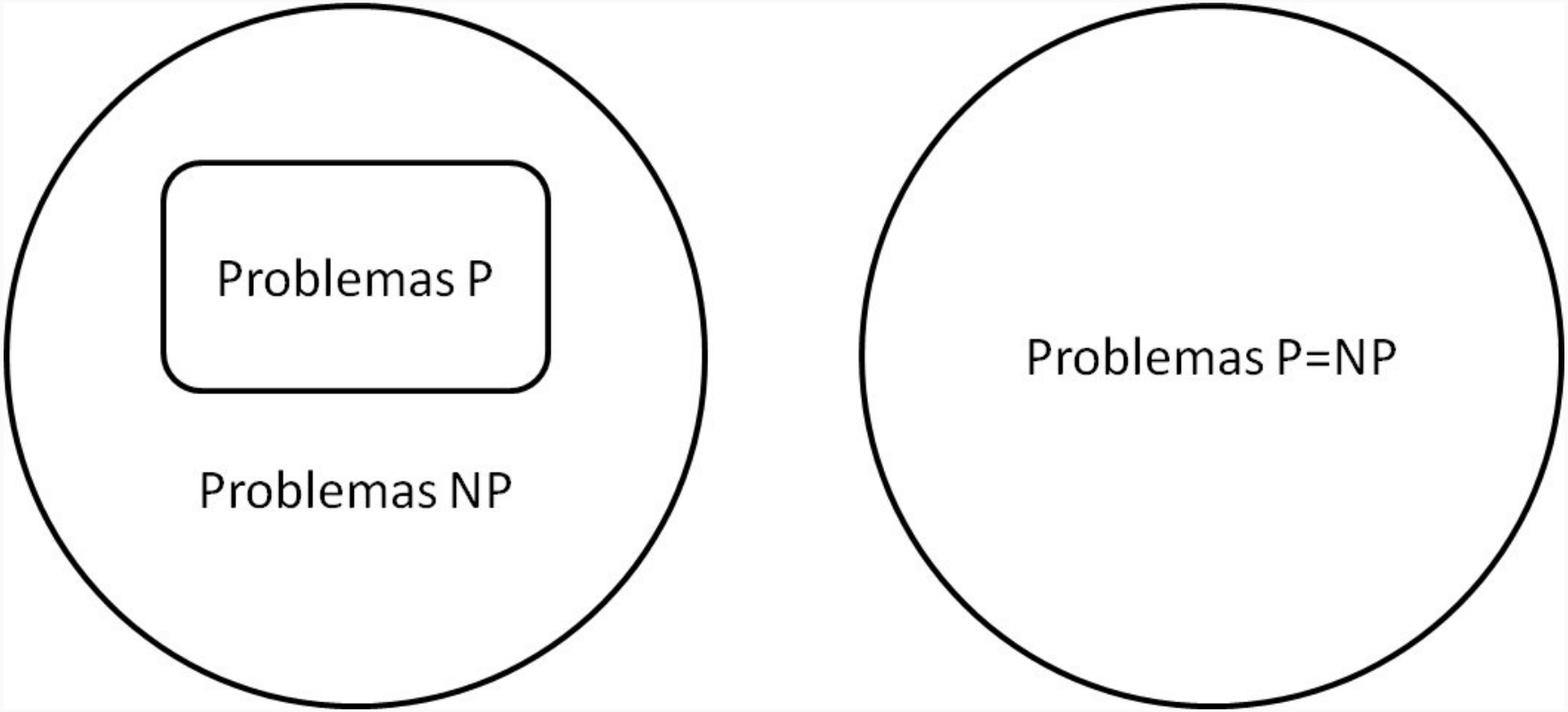

Figura 2: Posibles relaciones entre las clases P y NP

Relación entre clase P y NP

La evidencia teórica hasta el momento apunta hacia $P \subset NP$; sin embargo, no se ha demostrado que $P \neq NP$ [1, 2].

Una idea fundamental para establecer la relación entre los grupos P y NP es la **transformación polinomial** , también llamada **reducción polinomial**. La transformación polinomial de un lenguaje $L_1 \subseteq \Sigma_1*$ en un lenguaje $L_2 \subseteq \Sigma_2*$ es una función [5] $f : \Sigma_1* \rightarrow \Sigma_2*$ que satisface las siguientes condiciones [1, 2]:

- Existe un programa en una DM capaz de calcular la f en un tiempo polinomial
- Para todo $x \in \Sigma_1*$, $x \in L_1 \Leftrightarrow f(x) \in L_2$. Es decir, existe una regla de asociación entre L_1 y L_2 de tal forma que a cada elemento $x \in L_1$ se le asigna un único elemento $f(x) \in L_2$.

[5] Dados dos conjuntos A y B una función es una regla que asigna un elemento único $f(x) \in B$ a cada elemento $x \in A$

NP-Completos

Los problemas NP-Completos se definen como problemas muy difíciles en *NP* [9]. Se dice que un problema Π es NP-Completo si para cada problema $\Pi' \in NP$ existe una reducción en tiempo polinomial tal que $\Pi' \propto \Pi$. A continuación, se da la definición de NP-Completo.

Definición
Un problemas X es NP-Completo si $X \in NP$ y todo problema NP se reduce a X [2].

NP-Completos

El conjunto de los problemas NP-Completos es muy importante, pues hasta ahora cualquiera de estos problemas no puede ser solucionado de forma exacta por ningún algoritmo determinístico conocido en un tiempo polinomial; es decir, resolver un problema NP-completo en el peor caso es una tarea intratable [3]; ya que no es posible que un algoritmo exacto, encuentre la solución óptima con esfuerzos computacionales aceptables.

NP-Completos

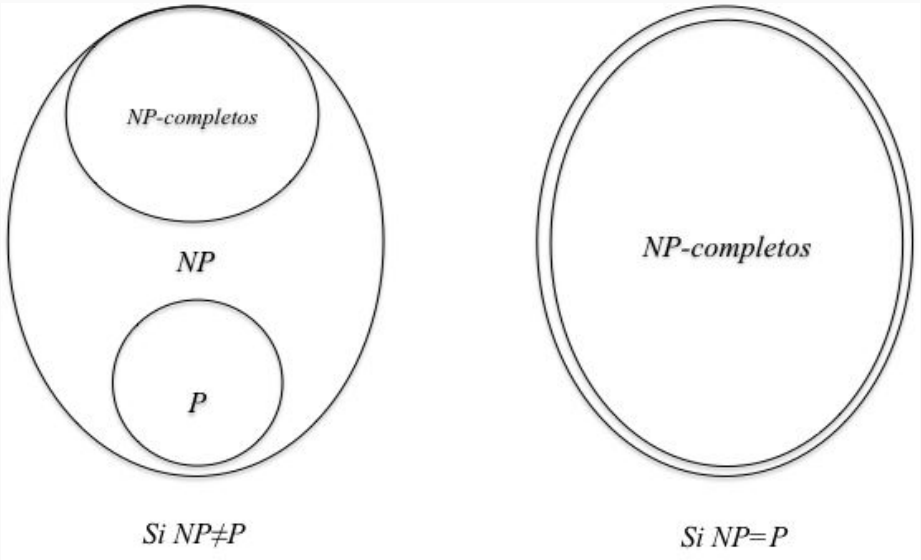

Figura 3: Posibles relaciones entre P, NP y NP-Completos Fuente [2]

NP-Completos

Teorema de Cook El teorema de Cook fue la primera prueba de existencia de problemas NP-Completos.

Teorema

El SAT es NP-Completo [4]

NP-Completos

El Teorema de Cook implica que $NP = P$, si y sólo si, el problema SAT es un P [2]. En otras palabras, el teorema de Cook dice que si SAT puede resolverse a través de algoritmo polinomial entonces todo problema NP puede ser resuelto en un número polinomial de pasos.

NP-Duros

Definición

Un problema de optimización o un problema de decisión Π *es llamado* $NP - Duro$ *si todo problema en NP pueden ser transformado polinomialmente a* Π *[8]. En otras palabras, dado un problema* Π *con lenguaje* L *y cualquier problema* $\Pi' \in NP$ *con lenguaje* L'*; si existe una transformación polinomial de* L' *a* L $(L' \propto L)$*, entonces se dice que* Π *es* $NP - Duro$*.*

NP-Duros

Cabe mencionar que no todos los problemas *NP* son *NP – Duros* ni todos los problemas *NP – Duros* son *NP* . En la Figura 4, se muestran la posible relacion entre *NP – Duro*, *NP – Completos* y *NP*.

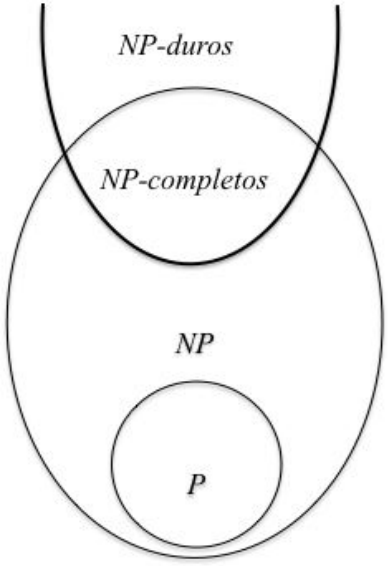

Figura 4: Conjeturas entre las relacion entre *NP*, *NP – Duros* y *NP – Completos* [10]

¿Cómo resolver un problema de programación matemática?

La resolución un problema es un proceso racional que involucra desde identificar el problema de interés hasta la elección y ejecución de alguna acción a fin de eliminarlo o reducirlo. Este proceso debe ser sistemático y guiado por el conocimiento disponible sobre el sistema.

Métodos de optimización

La tarea esencial de las herramientas de optimización consiste en localizar un punto meta (el mejor resultado con base al criterio de decisión) dentro del espacio de soluciones [11]. Lo cual, solamente se puede asegurar mediante el empleo de algoritmos exactos; sin embargo, cuando un problema es muy complejo no es posible implementar dichos algoritmos.

Por lo expuesto con anterioridad, se requiere realizar algún tipo de modificación (simplificación) en su formulación o bien se emplean alternativas de solución aproximada que resolverán el problema en un tiempo polinomial de forma casi-optima [3].

¿Por qué las heurísticas y las metaheurísticas?

Los métodos heurísticos y metaheurísticos surgen por la necesidad de resolver problemas con un alto grado de complejidad (NP-completos o NP-duros).

Definición

Las técnicas heurísticas son métodos o procedimientos para resolver un problema, que no son producto de un riguroso análisis formal. En la investigación de operaciones es un procedimiento para el que se tiene un alto grado de confianza en que encontrarán soluciones de alta calidad con un costo computacional razonable, aunque no garantiza optimalidad o factibilidad [12]

Metaheurísticas

Las técnicas metaheurísticas son estrategias inteligentes para alcanzar buenas soluciones a los problemas utilizando una cantidad razonable de recursos. Dichos procedimiento se caracterizan por poseer una fase de intensificación y otra de diversificación; estos métodos pueden aplicarse a una gran variedad de problema pues son métodos de propósito general; algunos ejemplos son: recocido simulado (SA), búsqueda tabú (TS), búsqueda en vecindades variables (SS), algoritmos bio-inspirados (Algoritmos Genéticos, Sistema inmune), algoritmos sociales (optimización por colonia de hormigas, optimización por nubes de partículas, algoritmo de sociedades y civilizaciones, algoritmos culturales), entre otros.

Metaheurísticas

La mayoría de las técnicas metaheurísticas tienen en común las siguientes características [13]:

- Son en alguna medida estocásticas, esta aproximación permite contener la explosión combinatoria.
- Generalmente tienen un origen en problemas discretos esto tiene ventajas para los problemas continuos.
- Se inspiran en analogías ya sean físicas (recocido simulado, difusión simulada) biológicas (algoritmos evolutivos, sistema inmune) etiológicas (colonia de hormigas, enjambre de partículas), memoria humana (redes neuronales) y procesos artísticos (búsqueda de la harmonía).

Metaheurísticas

A continuación se definen las propiedades deseables en las metaheurística [14]:

- **Simple**. La metaheurística debe estar basada en un principio sencillo y claro; fácil de comprender.
- **Precisa**. Los pasos y fases de la metaheurística deben estar formulados en términos concretos.
- **Coherente**. Los elementos de la metaheurística deben deducirse naturalmente de sus principios.
- **Eficaz**.Debe existir una alta probabilidad de alcanzar soluciones óptimas de casos realistas con la metaheurística.
- **Eficiente**. Se debe realizar un buen aprovechamiento de recursos computacionales: tiempo de ejecución y espacio de memoria.
- **General**. Debe ser utilizable con buen rendimiento en una amplia variedad de problemas.
- **Adaptable**. Debe ser capaz de adaptarse a diferentes contextos de aplicación o modificaciones importantes del modelo.
- **Robusta**. El comportamiento de la metaheurística debe ser poco sensible a pequeñas alteraciones del modelo o contexto de aplicación.
- **Interactiva**. Debe permitir que el usuario pueda aplicar sus conocimientos para mejorar el rendimiento del procedimiento.
- **Múltiple**. Debe suministrar diferentes soluciones alternativas de alta calidad entre las que el usuario pueda elegir.
- **Autónoma**. Debe permitir un funcionamiento autónomo, libre de parámetros o que se puedan establecer automáticamente.

Definición

Una metaheurística es un método de solución que organiza una interacción entre un procedimiento de búsqueda local y una estrategia de alto nivel (que permita escapar de los óptimos locales, así como robustecer el método de búsqueda en el espacio de soluciones) [15]. Es decir, las técnicas metaheurísticas son estrategias inteligentes para delinear mecanismos y mejoras a procedimientos heurísticos generales para resolver problemas con un alto rendimiento [12]

Razones por las cuales utilizar heurísticas o metaheurísticas

- No se conoce ningún método exacto para la resolución del problema.
- La complementación del mejor método exacto para la resolución del problema es muy costosa.
- Se requiere la alta flexibilidad que ofrecen los métodos heurísticos para la manipulación y resolución de problemas difíciles.
- Como parte de algún procedimiento de optimización:
  - Se usa para generar soluciones iniciales.
  - Se usa como un paso intermedio del procedimiento

Búsqueda Local

Stochastic Local Search

- Inicios: estudio de algoritmos de
  - Búsqueda Local en años 1950 (investigación de operaciones)
  - Búsqueda Local Estocástica en años 1960 (inteligencia artificial)

- Un área
  - en la intersección entre algoritmos, estadística, inteligencia artificial, investigación operativa, aplicaciones, ...
  - en desarrollo, adoptado para muchas aplicaciones

Generalidades

- Formulación del problema

$$\text{Min } f(s)$$

$$\text{s.t. } s \in \mathcal{S}$$

- La solución s es regularmente representada por variables de decisión, cuyo rango permitido de variación tiene que ser definido
- Solución $s \in \mathcal{S} =$ asignación de valores a las variables

Concepto de vecindario

Definición. Sea $(\mathcal{S}, f)$ una instancia de un problema de optimización combinatoria. Un vecindario es una función $\mathcal{N} : \mathcal{S} \mapsto 2^{\mathcal{S}}$ que, para cada solución $s \in \mathcal{S}$, define un conjunto de soluciones $\mathcal{N}(s)$ que están, en cierto sentido, cerca de s.

- Una solución $s' \in \mathcal{N}(s)$ es vecina de s.

Óptimo local

Definición. Sea $(\mathcal{S}, f)$ una instancia de un problema de optimización combinatoria y sea $\mathcal{N}$ una función de vecindario. Una solución $i \in \mathcal{S}$ es localmente mínima con respecto a $\mathcal{N}$ si $\forall j \in \mathcal{N}(i)$, $f(i) \leq f(j)$

- La optimalidad local depende por lo tanto de la función de vecindario utilizada

Vecindario exacto

Definición. Sea $(\mathcal{S}, f)$ una instancia de un problema de optimización combinatoria y sea $\mathcal{N}$ una función de vecindario. Se dice que $\mathcal{N}$ es exacto si los óptimos locales (de acuerdo a $\mathcal{N}$) también son óptimos globales.

- En caso de conocer un vecindario exacto, un simple algoritmo glotón (sin importar la modalidad de mejora) garantiza llegar a una solución óptima
- Implica, sin embargo, tiempos exponenciales para explorarlo

Referencias i


Michael R. Garey and David S. Jhonson.
Computers and intractability. A guide to the theory of NP-Completeness.
Freeman and company, 1979.


R. C. T. Lee, S. S. Tseng, and R. C. Chang.
Introducción al diseño y análisis de algoritmos. Un enfoque estratégico.
Mc Graw-Hill, 2007.


Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.
Introduction to algorithms.
MacGraw Hill Book company and The MIT Press, 2001.


Stephen Cook.
The p versus np problem,, 1992.

Referencias ii


Dina Goldin and Peter Wegner.
The church-turing thesis: Breaking the myth.
New Computational Paradigms, pages 152–168, 2005.


J.S.S. Román.
Lógica matemática y computabilidad.
Díaz de Santos, 1990.


Gonzalo Navarro.
Teoría de la computacón (lenguajes formales, computabilidad y complejidad), septimbre 2011.


Bernhard Korte and Jens Vygen.
Combinatorial optimization. Theory and algorithms.
Springer, 2000.

Referencias iii


W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver.
Combinatorial Optimization.
Wiley Series in Discrete Mathematics and Optimization. Wiley, 2011.


V.V. Muniswamy.
Design And Analysis Of Algorithms.
I.K. International Publishing House Pvt. Ltd., 2009.


William G Macready and David H. Wolpert.
What makes an optimization problem hard?
Complexity, 5:40–46, 1996.


Belén Melián, José A. Moreno Pérez, and J. Marcos Moreno Vega.
Metaheuristics: A global view.
Revista Iberoamericana de Inteligencia Artificial, 19:7–28, 2003.

Referencias iv


Johann Dréo, Alain Pétrowski, Patrick Siarry, and Eric Taillard.
Metaheuristics for hard optimization: methods and case studies.
Springer, 2006.


Rafael Martí.
Procedimientos metaheurísticos en optimiziaciín combinatoria.
Matemátiques, 1(1):3–62, 2003.


F. Glover and G.A. Kochenberger.
Handbook of metaheuristics.
International series in operations research & management science.
Kluwer Academic Publishers, 2003.