Teoría de la Complejidad Computacional

Tema 4: Clases de complejidad computacional. El problema P versus NP

David Orellana Martín

Grupo de Investigación en Computación Natural
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

dorellana@us.es

Máster Universitario en Matemáticas

Curso 2023-2024

Índice del tema IV

Complejidad respecto de una medida: problemática

Complejidad computacional inherente a un problema abstracto, respecto de una medida.

- Idea para una **definición local**:
  - ⋆ Se consideran todas las soluciones mecánicas del problema abstracto.
  - ⋆ Se calcula el coste de cada solución, respecto de la medida de complejidad considerada.
  - ⋆ Se elige una solución que use una cantidad mínima de recursos.
- **No es posible** una **definición local** (teorema de aceleración de Blum).
- Estudio de forma **global**, a través de las **clases de complejidad**.

Clases de Complejidad

Requisitos para la especificación de una **clase de complejidad**:

* Un **modelo de computación**.
* Un **modo de computación**.
* Los **recursos** a contabilizar (medida de complejidad).
* Una **cota superior** de los recursos permitidos.

Ejemplo:

* La clase de complejidad de los problemas resolubles por **MTD**s que usan una cantidad de recursos (por ejemplo, tiempo) de tipo polinomial.

Complejidad en tiempo de una MTD

Sea M una **MTD**.

Definición: Si $\mathcal{C} = (C_u, C_1, \ldots, C_t)$ es una computación de parada de M con entrada $u \in (\Sigma - \{B\})^*$, diremos que **t** es el **tiempo de ejecución** de $\mathcal{C}$.

Definición: M **trabaja en tiempo acotado por** una función total g de $\mathbb{N}$ en $\mathbb{N}$, **sii** existe $c > 0$ tal que para cada $u \in (\Sigma - \{B\})^{(n)}$, el tiempo de ejecución de M con entrada u es, a lo sumo, $c \cdot g(n)$ (asintóticamente: "**a partir de un lugar hacia adelante**").

(Se dirá que M es de **coste en tiempo acotado por** g)

Definición: La **complejidad en tiempo** de M es la función parcial t_M de $\mathbb{N}$ en $\mathbb{N}$ definida así:

* $t_M(n) =$ mayor tiempo de ejecución de M con entrada **cualquier** cadena de longitud n, en el caso en que todas las computaciones sean de parada.
* No definida, en caso contrario.

Complejidad en tiempo de una MTD

¿Cómo se interpreta el hecho de que $\boxed{t_M(13) = 97}$, siendo M una **MTD**?

- Puesto que $t_M(n) =$ mayor tiempo de ejecución de M con entrada **cualquier** cadena de longitud n, y $t_M(13) = 97$, se deduce lo siguiente:
  - **Todas las computaciones** de M con datos de entrada de tamaño 13, **han de ser de parada**.
  - Al ejecutar M con **cualquier** dato de **entrada** de **tamaño** 13, la máquina llegará a una **configuración de parada** en, **a lo sumo**, 97 pasos.
  - Para saber el resultado de M con cualquier entrada u de tamaño 13, será **suficiente simular** 97 de pasos de la computación $M(u)$.

Complejidad en tiempo de una MTD

Nota: Si $\{M_e : e \in \mathbb{N}\}$ es una **enumeración efectiva** de las **MTD**s, entonces se demuestra que $\{t_{M_e} : e \in \mathbb{N}\}$ es una medida de complejidad.

Definición: Una función $f : (\Sigma - \{B\})^* - \rightarrow \Sigma^*$ es **Turing–computable en tiempo acotado** por una función computable total 1-aria g, si existe una **MTD** que calcula la función f y trabaja en tiempo acotado por g.

Nota: Una función es **Turing–computable en tiempo polinomial** si es **Turing–computable** en tiempo acotado por un polinomio.

Codificaciones

Definición: Un conjunto A es **codificable** a partir de un conjunto B si existe una función total inyectiva y computable, f de A en B.

En tal situación se dirá que:

* f se es una **función codificadora** (o **codificación**) de A a partir de B.
* f^{-1} es la correspondiente **función decodificadora**.
* Para cada $x \in A$, $f(x) \in B$ es **el código** de x.

Ejemplos:

1. La función $f(x) = x^2$ es una codificación de $\mathbb{N}$ a partir de $\mathbb{N}$. La correspondiente función decodificadora es $f^{-1}(z) = \sqrt{z}$.
2. La función $f(x) = x^2$ **no** es una codificación de $\mathbb{Z}$ a partir de $\mathbb{N}$.

Codificaciones

Las **instancias** de problemas abstractos son cadenas sobre un alfabeto finito. Veamos que las instancias de cualquier problema se pueden codificar a partir de $\{0,1\}$.

Teorema: Sean $\Sigma = \{s_1, \ldots, s_n\}$ y $\Gamma = \{t_1, \ldots, t_r\}$ alfabetos **ordenados**, con $r \geq 2$, $s_1 < \ldots < s_n$ y $t_1 < \ldots < t_r$. Entonces, Σ^* es codificable a partir de Γ^* por una función, f, **computable** en **tiempo lineal** (existe $c \in \mathbb{N}$ con $|f(\delta)| = c \cdot |\delta|$, $\forall \delta \in \Sigma^*$).

Demostración:

Puesto que $r \geq 2$ existirá $p = \text{mín}\ \{k \in \mathbb{N} \mid |\Gamma^{(k)}| \geq n\}$. Se ordena alfabéticamente el conjunto $\{\alpha_1, \ldots, \alpha_q\}$ de cadenas de $\Gamma^{(p)}$ (entonces, $q \geq n$).

Sea g la aplicación inyectiva de Σ en Γ^* definida por $g(s_i) = \alpha_i$ y sea f la extensión natural de g a Σ^*; es decir, si $\delta = a_1 a_2 \ldots a_j$ es una cadena de Σ se define: $f(\delta) = g(a_1)g(a_2)\ldots g(a_j)$.

Entonces, f es una codificación de Σ^* en Γ^* y para cada $\delta = a_1 a_2 \ldots a_j \in \Gamma^*$: $|f(\delta)| = |g(a_1)g(a_2)\cdots g(a_j)| = |g(a_1)| + |g(a_2)| + \cdots + |g(a_j)| = p \cdot j = p \cdot |\delta|$. ■

Nota: Las **instancias** de problemas abstractos se pueden representar usando el alfabeto binario $\Sigma = \{0,1\}$, a partir de codificaciones *adecuadas*.

Decidibilidad (determinista) en tiempo polinomial

Definición: Una **MTD**, M, es de **coste en tiempo polinomial sii** trabaja en tiempo acotado por un polinomio (existe $k \in \mathbb{N}$ tal que $t_M(n) \in O(n^k)$).

Definición: Un lenguaje L es **decidible en tiempo polinomial** si existe una **MTD** de coste en tiempo polinomial que **decide** L.

Definición: Un **problema de decisión** es **resoluble algorítmicamente en tiempo polinomial** si existe una **MTD** de coste en tiempo polinomial que decide el lenguaje asociado al problema.

Clases de complejidad en tiempo

Sea f una función total computable 1-aria:

Definición: **TIME** (f) es el conjunto de todos los **problemas de decisión resolubles** por **MTD**s que trabajan en tiempo acotado por f.

La clase P:

$$\mathbf{P} = \bigcup_{k \in \mathbb{N}} \mathbf{TIME}\ (n^k)$$

P: conjunto de todos los **problemas de decisión resolubles** por **MTD**s que trabajan en tiempo polinomial.

P: clase de complejidad de los **problemas tratables** ("en modo determinista").

Decidibilidad no determinista en tiempo polinomial

Sea M una **MTND**:

* Diremos que M **decide un lenguaje** $L \subseteq \Sigma^*$ **sii** para cada $x \in \Sigma^*$ se verifica que $x \in L$ sii **existe, al menos, una computación de aceptación** de $M(x)$.

* Diremos que M **resuelve un problema de decisión** X **sii** M decide el lenguaje L_X asociado a dicho problema.

* Si $\mathcal{C} = (C_u, C_1, \ldots, C_t)$ es **una** computación de parada de M con entrada $u \in (\Sigma - \{B\})^*$, diremos que **t** es el **tiempo de ejecución** de $\mathcal{C}$.

Decidibilidad no determinista en tiempo polinomial

Sea M una **MTND**:

* La **complejidad en tiempo** de M es la función total $t_M : \mathbb{N} \to \mathbb{N}$ definida así:

$$t_M(n) = \text{máx}\{t'_M(x) : \ x \in \Sigma^* \wedge |x| = n\}$$

en donde $t'_M(x)$ se define como sigue:

  * $\min\{t(\mathcal{C}) \mid \mathcal{C}$ es computación de aceptación de $x\}$, si existe dicho mínimo.
  * 0, en caso contrario.

* M **trabaja en tiempo acotado por** $f(n)$ si existe $c > 0$ tal que $t_M(n) \leq c \cdot f(n)$ (asintóticamente: "**a partir de un lugar hacia adelante**").

* M es de **coste en tiempo polinomial** si trabaja en tiempo acotado por un polinomio.

Complejidad en tiempo de una MTND

¿Cómo se interpreta el hecho de que $\boxed{t_M(13) = 97}$, siendo M una **MTND**?

* Para calcular la complejidad en tiempo $t_M(13)$ en una **MTND**, M:
  * En primer lugar, se halla $t'_M(13)$; es decir, para cada cadena x de tamaño 13, se calcula $t'_M(x)$: el **menor tiempo** que tarda cualquier computación de aceptación de $M(x)$ (si no hay ninguna, ese valor será 0).
  * Seguidamente, se calcula el **mayor** de los valores $t'_M(x)$ obtenidos en el paso anterior.
* Luego, al ejecutar, **a lo sumo**, 97 de pasos de todas las computaciones de M con entrada cualquier cadena u de tamaño 13, detectaríamos si existe alguna computación de aceptación de $M(u)$.
* Por tanto:
  * Si tras ejecutar, **a lo sumo**, **97 pasos** de **todas** las **computaciones** de M con **entrada** cualquier dato de **tamaño 13**, no se encuentra ninguna que sea de aceptación, entonces **podemos afirmar** que **no existe** una tal computación, por muchos pasos que se simulen.

La clase NP

Sea f una función total computable 1-aria:

Definición: **NTIME** (f) es el conjunto de todos los **problemas de decisión resolubles** por **MTND**s que trabajan en tiempo acotado por f.

* Obviamente, **TIME**$(f) \subseteq$ **NTIME** (f).

La clase NP:

$$\mathbf{NP} = \bigcup_{k \in \mathbb{N}} \mathbf{NTIME}\ (n^k)$$

NP: conjunto de todos los **problemas de decisión resolubles** por **MTND**s que trabajan en tiempo polinomial.

NP: clase de complejidad de los **problemas tratables** ("en modo no determinista").

El problema P versus NP

- Se tiene que **P** $\subseteq$ **NP**: ¿Es estricta la inclusión **P**$\subseteq$**NP**?

- Problema **P versus NP**: determinar si coinciden las clases **P** y **NP**.
- El problema **P versus NP** es uno de los **siete problemas del milenio** (Clay Mathematics Institute: The Millenium Prize Problems)

La conjetura P $\neq$ NP

- Informalmente, dicho problema consiste en determinar **qué es más difícil**:
  - **Hallar la solución** de un problema **o comprobar** si una presunta solución es, realmente, una solución correcta.

  Se cree que **resolver** un problema debería ser más difícil que **chequear** la corrección de una presunta solución del mismo.

  Esta creencia generalizada, se puede ilustrar con un ejemplo:

  - Dado un sistema lineal de 3 ecuaciones lineales con 3 incógnitas,
    - "Parece" que **hallar la solución** del sistema (una terna de números, infinitas ternas o ninguna) ha de ser *estrictamente más difícil* que **comprobar** si tres números concretos satisfacen todas las ecuaciones del sistema.

  Lo que se expresa diciendo: **"todo parece indicar"** que **P** $\neq$ **NP**.

La conjetura P $\neq$ NP

¿Cómo se abordaría la resolución de dicha conjetura?

- Definición formal del concepto: un problema abstracto es **más dificil** que otro (**reducibilidad en tiempo polinomial**).
- Si se quiere probar que **P** $\subsetneq$ **NP**, los candidatos idóneos serían los problemas **más difíciles** de **NP** (denominados **problemas NP-completos**).
- En 1970, S. Cook demuestra la **NP**-completitud de un problema relacionado con la **lógica** proposicional.
- Metodologia para probar que **P** $\neq$ **NP**:
  - Elegir **un** problema **NP**-completo y probar que **no** pertenece a **P**.
  - Ahora bien, si se prueba que **un** problema **NP**-completo **pertenece** a **P** entonces resultaría que **P**=**NP**.

Se cree que **P** $\neq$ **NP** y, por ello, se dice que los problemas **NP**-completos son **presuntamente intratables**.