Complejidad Parametrizada (Parte 1)

Alejandro Grez

Pontificia Universidad Católica

ajgrez@uc.cl

7 de Junio, 2016

Overview

1. Intuición
2. Clase FPT
   - Reducciones fpt
3. Clase para-NP
4. Clase XP

Intuición

Las clases de complejidad comunes analizan y clasifican problemas de acuerdo a la cantidad de un recurso (e.g. espacio, tiempo) que requieren los agoritmos para resolverlos.

La cantidad del recurso generalmente es medida como una función sobre el tamaño del input.

Pero generalmente los problemas cumplen ciertas estructuras, y se pueden definir parámetros en el input.

El objetivo es analizar la complejidad de problemas en los que sabemos que algún parámetro será considerablemente pequeño (con respecto al input).

Intuición

Sabemos que SAT es un problema dificil (NP-completo), pero ¿y si supieramos que la cantidad de variables es pequeña?

El problema de evaluar una consulta φ sobre una base de datos B es dificil (NP-hard), pero ¿y si consideramos que φ es mucho más pequeño que B? (usualmente el input de este problema es de la forma (φ, B))

Lenguaje Parametrizado

Consideremos el alfabeto Σ.

Una parametrización $\kappa : \Sigma^* \to \mathbb{N}$ es una función computable en tiempo polinomial.

Un lenguage parametrizado es un par (L, κ), donde $L \subseteq \Sigma^*$ y κ es una parametrización.

- e.g. (SAT, κ) con $\kappa(\varphi) = |var(\varphi)|$

Fixed-Parameter Tractable

Una MT M es *fixed-parameter tractable* con respecto a κ (κ-fpt) si existe una función computable $f : \mathbb{N} \to \mathbb{N}$ y un polinomio p tal que el tiempo de ejecución de M con input w es a lo más $f(\kappa(w)) \cdot p(|w|)$.

Un lenguaje parametrizado (L, κ) es *fixed-parameter tractable* (fpt) si existe una MT M que sea κ-fpt y tal que $L(M) = L$.

- Cada vez que hablemos de un lenguaje, tendrá asignada una parametrización κ.

Definimos la clase FPT como el conjunto de todos los lenguajes que sean fpt.

SAT y FPT

Consideremos la parametrización $\kappa(\varphi) = |var(\varphi)|$, en donde φ es una fórmula en lógica proposicional (si φ no corresponde a una fórmula en LP, se puede definir $\kappa(\varphi) = 1$).

Ahora consideremos el lenguaje (SAT, κ). ¿$(SAT, \kappa) \in$ FPT?

Idea para κ

Nótese que, para cualquier lenguaje decidible L, existe una parametrización κ tal que $(L, \kappa) \in \text{FPT}$.

- Considere $\kappa(w) = |w|$.

El objetivo de κ es representar un parámetro que sea considerablemente pequeño en comparación con el input.

κ es considerado como parte del problema (y parte del análisis de complejidad de éste).

Slices (rebanadas)

Dado un lenguaje (L, κ), definimos la i-slice de (L, κ) como el lenguaje:

$$\{w \mid w \in L \wedge \kappa(w) = i\}$$

- e.g. la 10-slice de (SAT,κ) son las formulas satisfacibles con 10 variables.

Si $(L, \kappa) \in$ FPT, entonces, para cualquier i, la i-slice de (L, κ) esta en PTIME.

Consideremos el problema COLORABILITY y la parametrización $\kappa((G, k)) = k$.

- ¿Qué pasa si (COLORABILITY, κ) $\in$ FPT?

Reducción fpt

Consideremos los lenguajes parametrizados (L_1, κ_1) y (L_2, κ_2). Un mapeo $R : \Sigma^* \to \Sigma^*$ se dice que es reducción-fpt de (L_1, κ_1) a (L_2, κ_2) si:

1. Para todo $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow R(w) \in L_2$
2. R es computable por una κ-fpt MT. Es decir, existe una función computable f y un polinomio p tal que $R(w)$ es computable en tiempo $f(\kappa_1(w)) \cdot p(|w|)$.
3. Existe una función computable $g : \mathbb{N} \to \mathbb{N}$ tal que $\kappa_2(R(w)) \leq g(\kappa_1(w))$ para todo $w \in \Sigma^*$
   - Pronto veremos para qué sirve esta condición

Si existe dicha reducción, diremos que (L_1, κ_1) es fpt-reducible a (L_2, κ_2), o $(L_1, \kappa_1) \leq^{fpt} (L_2, \kappa_2)$

Reducción fpt

FPT es cerrada bajo reducción-fpt. Esto es, si $(L_1, \kappa_1) \leq^{fpt} (L_2, \kappa_2)$ y $(L_2, \kappa_2) \in \text{FPT}$, entonces $(L_1, \kappa_1) \in \text{FPT}$.

- Demostración en pizarra (acá usaremos la condición 3).

Reducción fpt

Consideremos un lenguaje cualquiera en FPT, e.g. $(L_1, \kappa_1) \in \text{FPT}$. Para cualquier lenguaje no trivial (L_2, κ_2), se cumple que $(L_1, \kappa_1) \leq^{fpt} (L_2, \kappa_2)$.

- Demostración en pizarra (análogo a PTIME y $\leq_p^m$).

¿No-determinismo?

¿Qué pasaría si agregamos no-determinismo a M?

para-NP

Una MT *no determinista* M es κ-fpt si existe una función computable $f : \mathbb{N} \to \mathbb{N}$ y un polinomio p tal que el tiempo de ejecución de M con input w es a lo más $f(\kappa(w)) \cdot p(|w|)$.

Un lenguaje parametrizado (L, κ) pertenece a para-NP si existe una MT no-determinista M que sea κ-fpt y tal que $L(M) = L$

para-NP

Dado un $L \in$ NP, se cumple que $(L, \kappa) \in$ para-NP **para cualquier** κ.

En particular, sabemos que COLORABILITY $\in$ NP.

- Por lo tanto, si definimos $\kappa((G, k)) = k$, se cumple que (COLORABILITY, κ) $\in$ para-NP.

¿FPT=para-NP?

- Se cree que no. ¿Por qué?

XP

Un lenguaje (L, κ) pertenece a XP si existe una función computable $f : \mathbb{N} \to \mathbb{N}$ y una MT M que con input w decide si $w \in L$ en tiempo

$$|w|^{f(\kappa(w))} + f(\kappa(w))$$

- ¿Qué complejidad tendría el lenguaje de la i-slice de (L, κ)?

Es claro que FPT $\subseteq$ XP.

- ¿Puede ser que FPT $=$ XP?

XP y p-EXP-DTM-HALT

Definamos el lenguaje p-EXP-DTM-HALT como las tuplas $((M, n, k), \kappa)$, en que $\kappa(M, n, k) = k)$, k es un número natural, n es un número natural en unario y M es una MT que acepta el string vacío en tiempo n^k.

p-EXP-DTM-HALT es XP-completo.

- Demostrar p-EXP-DTM-HALT $\in$ XP es trivial.
- ¿Y XP-hardness?

p-EXP-DTM-HALT es XP-hard

Daremos una reducción fpt de un lenguaje cualquiera $(L, \kappa) \in$ XP a p-EXP-DTM-HALT.

Sea una función computable $f : \mathbb{N} \to \mathbb{N}$ y una MT M que decide si un input $w \in L$ en tiempo $|w|^{f(\kappa(w))} + f(\kappa(w))$.

Asumimos que $f(\kappa(w)) < |w|^{f(\kappa(w))}$, por lo que $|w|^{f(k(w))} + f(k(w)) < 2|w|^{f(k(w))} < |w|^{g(k(w))}$ para algún g computable.

Dado un input w:

1. Definimos la MT M_w que anota w en su cinta de input y simula M con input w.
2. Definimos $n_w = |w|$
3. Definimos $k_w = g(k(w))$
4. Retornamos (M_w, n_w, k_w)

FPT $\subset$ XP

Demostración:

1. Por contradicción, asumamos que FPT=XP
2. En particular, p-EXP-DTM-HALT $\in$ FPT
3. Existe una κ-fpt MT M_1, una función f_1 y un polinomio p_1 tales que M_1 decide si $((M, n, k), \kappa) \in p$-EXP-DTM-HALT en tiempo $f_1(\kappa(w)) \cdot p_1(|w|)$. Consideremos $p_1(x) = x^c$ para algún c
4. Consideremos la $(c+1)$-slice. Esta es equivalente al problema de, dada una MT M y un entero n en unario, ver si M acepta el string vacío en n^{c+1} pasos
5. $\kappa(w) = c+1$, por lo que $f_1(\kappa(w)) = f_1(c+1)$ es una constante. Para n suficientemente grande, e.g. $n \geq |M| + |k|$, $|(M, n, k)| \leq 2 \cdot n$, por lo que $p_1(|w|) = |w|^c$ es $O(n^c)$
6. 4 define un lenguaje en DTIME(n^{c+1}), y 5 dice que hay un algoritmo para resolverlo en DTIME(n^c). Esto implica que DTIME(n^{c+1}) $\subseteq$ DTIME(n^c), lo que contradice el teorema de jerarquía de tiempo.

References

Flum, J., Grohe, M (2006)

Parameterized Complexity Theory (Libro)

Ugarte, M. (2012)

An introduction to Parameterized Complexity Theory (Diapositivas)

FIN