Clases de complejidad de funciones

IIC3810

Marcelo Arenas y Luis Alberto Croquevielle

Calculando una función

Dado un alfabeto Σ, consideramos funciones $f : \Sigma^* \to \mathbb{N}$

Representamos cada número natural como un string sobre el alfabeto $\{0, \ldots, 9\}$

- Si una MTC calcula una función $f : \Sigma^* \to \mathbb{N}$, entonces para cada $w \in \Sigma^*$ suponemos que $f(w)$ es retornado en la cinta de salida como un string sobre el alfabeto $\{0, \ldots, 9\}$

Las funciones computables en tiempo polinomial

Definición

Una función $f : \Sigma^* \to \mathbb{N}$ está en la clase FP si y sólo si existe una MTC que funciona en tiempo polinomial y calcula f

¿Qué funciones son difíciles de computar?

- Consideramos funciones que si son computables en tiempo polinomial nos dan algoritmos polinomiales para resolver problemas NP-completos

La clase de funciones #P

Sea M una MT no determinista con alfabeto de entrada Σ

Supuesto

Si decimos que M funciona en tiempo polinomial, entonces suponemos que existe un polinomio $p(n)$ tal que para cada $w \in \Sigma^*$ y cada ejecución de M con entrada w, el número de pasos en la ejecución es menor o igual a $p(|w|)$

- ¿Por qué podemos realizar este supuesto sin perdida de generalidad?

Dado $w \in \Sigma^*$, definimos $\text{accept}_M(w)$ como el número de ejecuciones de M con entrada w que se detienen en un estado final.

- En particular, $w \in L(M)$ si y sólo si $\text{accept}_M(w) > 0$

La clase de funciones #P

Definición

Una función $f : \Sigma^* \to \mathbb{N}$ *está en* $\#P$ *si y sólo si existe una MT no determinista* M *con alfabeto de entrada* Σ*, que funciona en tiempo polinomial y tal que para cada* $w \in \Sigma^*$*:* $f(w) = accept_M(w)$

Ejercicio

Sea #SAT una función tal que, dada una fórmula proposicional φ, retorna el número de valuaciones que satisfacen φ. Demuestre que #SAT $\in$ #P

Para pensar . . .

Ejercicios

1. Demuestre que si $f, g \in \#\mathrm{P}$, entonces $f + 1 \in \#\mathrm{P}$ y $f + g \in \#\mathrm{P}$

2. Demuestre que si $f, g \in \#\mathrm{P}$, entonces $f \cdot g \in \#\mathrm{P}$

3. Demuestre que $\mathrm{FP} \subseteq \#\mathrm{P}$

Una primera noción de reducción para funciones

Definición

Dadas funciones $f, g : \Sigma^* \to \mathbb{N}$, *decimos que* f *se puede reducir a* g *de forma parsimoniosa, denotado como* $f \leq^p_{par} g$, *si existe una función* $h : \Sigma^* \to \Sigma^*$ *tal que* h *puede ser calculada en tiempo polinomial y para todo* $w \in \Sigma^*$:

$$f(w) \quad = \quad g(h(w))$$

Ejercicio

Demuestre que si $f_1 \leq^p_{par} f_2$ y $f_2 \leq^p_{par} f_3$, entonces $f_1 \leq^p_{par} f_3$

Una noción de reducción más general para funciones

Al igual que para las máquinas de Turing con oráculos para problemas de decisión, podemos definir las máquinas de Turing con oráculos para funciones

- ¿Cómo debería funcionar la cinta de consulta en este caso?

Dada una función $f : \Sigma^* \to \mathbb{N}$, denotamos como M^f a una MT que tiene a f como oráculo

- M^f también puede ser una MTC

Una noción de reducción más general para funciones

Dadas funciones $f, g : \Sigma^* \to \mathbb{N}$, decimos que $f \in \mathrm{FP}^g$ si existe una MTC M^g que funciona en tiempo polinomial y calcula f

Definición

Dadas funciones $f, g : \Sigma^* \to \mathbb{N}$, decimos que f se puede reducir a g en tiempo polinomial, denotado como $f \leq_T^p g$, si $f \in FP^g$

Ejercicio

Demuestre que si $f_1 \leq_T^p f_2$ y $f_2 \leq_T^p f_3$, entonces $f_1 \leq_T^p f_3$

Una noción de completitud para #P

Definición

Una función $f : \Sigma^* \to \mathbb{N}$ es #P-hard si para cada $g \in \#P$ se tiene que $g \leq_T^p f$. Sí adicionalmente $f \in \#P$, entonces f se dice #P-completo.

Si $\leq_T^p$ es reemplazado por $\leq_{par}^p$ en la definición anterior, entonces decimos que f es #P-hard o #P-completo bajo reducciones parsimoniosas

- Si f es #P-hard bajo reducciones parsimoniosas entonces también es #P-hard, puesto que $f \leq_{par}^p g$ implica $f \leq_T^p g$

Un primer problema #P-completo

Teorema

#SAT es #P-completo bajo reducciones parsimoniosas.

Ejercicio

Demuestre el teorema.

Un segundo problema #P-completo

Sea #CNF-SAT una función que, dada una formula proposicional φ en CNF, retorna el número de valuaciones que satisfacen φ

Teorema

#CNF-SAT es #P-completo bajo reducciones parsimoniosas.

Ejercicio

Demuestre el teorema.

Un tercer problema #P-completo

Sea #DNF-SAT una función que, dada una formula proposicional φ en DNF, retorna el número de valuaciones que satisface φ

Teorema

#DNF-SAT es #P-completo

Demostración: vamos a demostrar que #CNF-SAT $\leq_T^p$ #DNF-SAT

Sea φ una fórmula proposicional en CNF, y sea ψ una fórmula proposicional en DNF que es equivalente a $\neg\varphi$

- ψ puede ser construida en tiempo polinomial a partir de φ

La demostración del teorema

Definimos nv$(\cdot)$ como una función que retorna el número de variables de una fórmula proposicional φ

- Por ejemplo, $\text{nv}((p \vee q) \wedge \neg r) = 3$ y $\text{nv}((r \vee q) \wedge \neg r) = 2$

La función nv$(\cdot)$ es computable en tiempo polinomial

Tenemos que:

$$\#\text{CNF-SAT}(\varphi) \quad = \quad 2^{\text{nv}(\psi)} - \#\text{DNF-SAT}(\psi)$$

Concluimos entonces que $\#\text{CNF-SAT} \in \text{FP}^{\#\text{DNF-SAT}}$

- Por lo tanto, $\#\text{CNF-SAT} \leq_T^p \#\text{DNF-SAT}$ □

Sobre la completitud de #DNF-SAT

¿Se puede demostrar que #DNF-SAT es #P-completo bajo reducciones parsimoniosas?

Para responder esta pregunta, defina para cada función $f : \Sigma^* \to \mathbb{N}$ el siguiente problema de decisión:

$$L_f \quad = \quad \{w \in \Sigma^* \mid f(w) > 0\}$$

Ejemplo

Tenemos que $L_{\#\text{SAT}} = \text{SAT}$

Sobre la completitud de #DNF-SAT

Proposición

Si $f \leq^p_{par} g$ y $L_g \in PTIME$, entonces $L_f \in PTIME$

Ejercicio

Demuestre la proposición.

Concluimos que #DNF-SAT no puede ser #P-completo bajo reducciones parsimoniosas, a menos que PTIME = NP

- Puesto que si #SAT $\leq^p_{par}$ #DNF-SAT, entonces SAT $\in$ PTIME

¿Cómo podemos calcular una función #P-completa?

Sea $f : \Sigma^* \to \mathbb{N}$ una función en #P

Si la función f es #P-completa entonces no esperamos tener un algoritmo polinomial para calcularla.

Pero el hecho de que f sea #P-completa no implica que el valor de f no pueda ser aproximado de manera eficiente.

- Vamos a estudiar una noción de aproximación aleatorizada para funciones en #P

Aproximación aleatorizada de funciones

Sea $f : \Sigma^* \to \mathbb{N}$

- Y sea $\mathbb{R}^+$ el conjunto de los números reales mayores a 0

Definición

Un algoritmo aleatorizado $\mathcal{A} : \Sigma^* \times \mathbb{R}^+ \to \mathbb{N}$ *es un* **fully polynomial randomized approximation scheme (FPRAS)** *para* f *si existe un polinomio* $p(x, y)$ *tal que para cada* $w \in \Sigma^*$ *y* $\varepsilon \in \mathbb{R}^+$*:*

1. *El número de pasos ejecutados por* $\mathcal{A}(w, \varepsilon)$ *es menor o igual a* $p(|w|, \frac{1}{\varepsilon})$
2. $\mathbf{Pr}(|\mathcal{A}(w, \varepsilon) - f(w)| \leq \varepsilon \cdot f(w)) \geq \frac{3}{4}$

Un enfoque general para construir un FPRAS

Dada una función $f : \Sigma^* \to \mathbb{N}$, $w \in \Sigma^*$ y $\varepsilon \in \mathbb{R}^+$, definimos una variable aleatoria X tal que $\mathbf{E}[X] = f(w)$

- Para tener una estimación de $f(w)$ nos basta muestrear la variable aleatoria X

Debe ser posible muestrear X en tiempo polinomial en $|w|$ y $\frac{1}{\varepsilon}$, y además debemos tener que $\mathbf{Pr}(|X - f(w)| \leq \varepsilon \cdot f(w)) \geq \frac{3}{4}$

- Debemos entonces acotar inferiormente $\mathbf{Pr}(|X - \mathbf{E}[X]| \leq \varepsilon \cdot \mathbf{E}[X])$

Vamos a estudiar entonces algunas desigualdades útiles para obtener una cota inferior para $\mathbf{Pr}(|X - \mathbf{E}[X]| \leq \varepsilon \cdot \mathbf{E}[X])$

La desigualdad de Markov

Teorema

Sea X una variable aleatoria no negativa. Para cada $a \in \mathbb{R}^+$ se tiene que:

$$\mathbf{Pr}(X \geq a) \quad \leq \quad \frac{\mathbf{E}[X]}{a}$$

Una demostración de la desigualdad de Markov

Suponemos que el recorrido de X es un conjunto finito $\Omega \subseteq \mathbb{R}_0^+$:

$$
\begin{aligned}
\mathbf{E}(X) &= \sum_{r \in \Omega} r \cdot \mathbf{Pr}(X = r) \\
&= \left(\sum_{r \in \Omega \,:\, r < a} r \cdot \mathbf{Pr}(X = r) \right) + \left(\sum_{s \in \Omega \,:\, s \geq a} s \cdot \mathbf{Pr}(X = s) \right) \\
&\geq \sum_{s \in \Omega \,:\, s \geq a} s \cdot \mathbf{Pr}(X = s) \\
&\geq \sum_{s \in \Omega \,:\, s \geq a} a \cdot \mathbf{Pr}(X = s) \\
&= a \cdot \left(\sum_{s \in \Omega \,:\, s \geq a} \mathbf{Pr}(X = s) \right) \\
&= a \cdot \mathbf{Pr}(X \geq a)
\end{aligned}
$$

Concluimos que $\mathbf{Pr}(X \geq a) \leq \frac{\mathbf{E}(X)}{a}$ □

La desigualdad de Chebyshev

Teorema

$$\mathbf{Pr}(|X - \mathbf{E}[X]| \geq a) \quad \leq \quad \frac{\mathbf{Var}[X]}{a^2}$$

Demostración. Utilizando la desigualdad de Markov obtenemos:

$$\begin{aligned} \mathbf{Pr}(|X - \mathbf{E}[X]| \geq a) &= \mathbf{Pr}((X - \mathbf{E}[X])^2 \geq a^2) \\ &\leq \frac{\mathbf{E}[(X - \mathbf{E}[X])^2]}{a^2} \\ &= \frac{\mathbf{Var}[X]}{a^2} \end{aligned}$$

□

Utilizando la desigualdad de Chebyshev

A partir de la desigualdad de Chebyshev obtenemos:

$$\mathbf{Pr}(|X - \mathbf{E}[X]| \leq \varepsilon \cdot \mathbf{E}[X]) \quad \geq \quad 1 - \frac{\mathbf{Var}[X]}{\varepsilon^2 \cdot \mathbf{E}[X]^2}$$

Es posible mejorar esta cota inferior considerando el promedio de $k \geq 2$ muestras de X obtenidas de manera independiente.

- Así reducimos el impacto de las muestras alejadas de $\mathbf{E}[X]$

Utilizando la desigualdad de Chebyshev

Por la desigualdad de Chebyshev:

$$\mathbf{Pr}\left(\left|\frac{1}{k}\cdot\sum_{i=1}^{k}X_i - \mathbf{E}\left[\frac{1}{k}\cdot\sum_{i=1}^{k}X_i\right]\right| \geq \varepsilon\cdot\mathbf{E}\left[\frac{1}{k}\cdot\sum_{i=1}^{k}X_i\right]\right) \leq \frac{\mathbf{Var}\left[\frac{1}{k}\cdot\sum_{i=1}^{k}X_i\right]}{\varepsilon^2\cdot\mathbf{E}\left[\frac{1}{k}\cdot\sum_{i=1}^{k}X_i\right]^2}$$

Además, tenemos que:

$$\begin{aligned}
\mathbf{E}\left[\frac{1}{k}\cdot\sum_{i=1}^{k}X_i\right] &= \frac{1}{k}\cdot\mathbf{E}\left[\sum_{i=1}^{k}X_i\right] \\
&= \frac{1}{k}\cdot\sum_{i=1}^{k}\mathbf{E}[X_i] \\
&= \frac{1}{k}\cdot\sum_{i=1}^{k}\mathbf{E}[X] \\
&= \mathbf{E}[X]
\end{aligned}$$

Utilizando la desigualdad de Chebyshev

$$\begin{aligned}
\mathbf{Var}\left[\frac{1}{k}\cdot\sum_{i=1}^{k}X_i\right] &= \frac{1}{k^2}\cdot\mathbf{Var}\left[\sum_{i=1}^{k}X_i\right] \\
&= \frac{1}{k^2}\cdot\sum_{i=1}^{k}\mathbf{Var}[X_i] \\
&= \frac{1}{k^2}\cdot\sum_{i=1}^{k}\mathbf{Var}[X] \\
&= \frac{1}{k}\cdot\mathbf{Var}[X]
\end{aligned}$$

De lo cual concluimos que:

$$\mathbf{Pr}\left(\left|\frac{1}{k}\cdot\sum_{i=1}^{k}X_i-\mathbf{E}[X]\right|\geq\varepsilon\cdot\mathbf{E}[X]\right) \leq \frac{\mathbf{Var}[X]}{k\cdot\varepsilon^2\cdot\mathbf{E}[X]^2}$$

Utilizando la desigualdad de Chebyshev

Realizando k muestras independientes de la variable aleatoria X obtenemos entonces:

$$\mathbf{Pr}\left(\left|\frac{1}{k}\cdot\sum_{i=1}^{k}X_i - \mathbf{E}[X]\right| \leq \varepsilon\cdot\mathbf{E}[X]\right) \quad \geq \quad 1 - \frac{\mathbf{Var}[X]}{k\cdot\varepsilon^2\cdot\mathbf{E}[X]^2}$$

Podemos entonces ajustar el valor de k para obtener un FPRAS.

- El valor de $\mathbf{Var}[X]$ debe ser polinomial en el tamaño de la entrada para obtener un FPRAS.

Construyendo un FPRAS para #DNF-SAT

Sea $\varphi = D_1 \vee \cdots \vee D_m$ una formula proposicional en DNF

- Cada D_i es una conjunción de literales
- φ menciona n variables proposicionales, vale decir, $\text{nv}(\varphi) = n$

Suponemos que cada D_i no tiene literales repetidos ni complementarios

- ¿Por qué podemos suponer esto?
- Sea ℓ_i el número de literales mencionados en D_i

Un FPRAS para #DNF-SAT: primer intento

Suponga que las asignaciones de valores de verdad σ para φ son escogidas con distribución uniforme:

$$\mathbf{Pr}(\sigma) \quad = \quad \frac{1}{2^n}$$

Y considere la siguiente variable aleatoria:

$$X(\sigma) \quad = \quad \begin{cases} 2^n & \sigma(\varphi) = 1 \\ 0 & \sigma(\varphi) = 0 \end{cases}$$

Un FPRAS para #DNF-SAT: primer intento

Tenemos que:

$$\begin{aligned}
\mathbf{E}[X] &= \sum_{\sigma} X(\sigma) \cdot \mathbf{Pr}(\sigma) \\
&= \sum_{\sigma \,:\, \sigma(\varphi)=1} 2^n \cdot \frac{1}{2^n} \\
&= \sum_{\sigma \,:\, \sigma(\varphi)=1} 1 \\
&= \#\text{DNF-SAT}(\varphi)
\end{aligned}$$

Tenemos entonces que $\mathbf{E}[X] = \#\text{DNF-SAT}(\varphi)$, el primer requisito para la variable aleatoria que queremos construir

- Además se puede muestrear X en tiempo polinomial en n
  - ¿Cómo se hace esto?

Un FPRAS para #DNF-SAT: primer intento

Nos falta calcular $\mathbf{Var}[X]$ para saber si podemos obtener un FPRAS.

Tenemos que:

$$\begin{aligned}
\mathbf{E}[X^2] &= \sum_{\sigma} X^2(\sigma) \cdot \mathbf{Pr}(\sigma) \\
&= \sum_{\sigma \,:\, \sigma(\varphi)=1} 2^{2 \cdot n} \cdot \frac{1}{2^n} \\
&= \sum_{\sigma \,:\, \sigma(\varphi)=1} 2^n \\
&= 2^n \cdot \#\text{DNF-SAT}(\varphi)
\end{aligned}$$

Por lo tanto:

$$\begin{aligned}
\mathbf{Var}[X] &= \mathbf{E}[X^2] - \mathbf{E}[X]^2 \\
&= 2^n \cdot \#\text{DNF-SAT}(\varphi) - \#\text{DNF-SAT}(\varphi)^2 \\
&= (2^n - \#\text{DNF-SAT}(\varphi)) \cdot \#\text{DNF-SAT}(\varphi)
\end{aligned}$$

Un FPRAS para #DNF-SAT: primer intento

Realizando k muestras independientes de X obtenemos entonces:

$$\begin{aligned}\mathbf{Pr}\left(\left|\frac{1}{k}\cdot\sum_{i=1}^{k}X_i-\mathbf{E}[X]\right|\geq\varepsilon\cdot\mathbf{E}[X]\right) &\leq \frac{\mathbf{Var}[X]}{k\cdot\varepsilon^2\cdot\mathbf{E}[X]^2}\\ &= \frac{(2^n-\#\text{DNF-SAT}(\varphi))\cdot\#\text{DNF-SAT}(\varphi)}{k\cdot\varepsilon^2\cdot\#\text{DNF-SAT}(\varphi)^2}\\ &= \frac{2^n-\#\text{DNF-SAT}(\varphi)}{k\cdot\varepsilon^2\cdot\#\text{DNF-SAT}(\varphi)}\end{aligned}$$

Por lo tanto, para obtener $\mathbf{Pr}(|\frac{1}{k}\cdot\sum_{i=1}^{k}X_i-\mathbf{E}[X]|\geq\varepsilon\cdot\mathbf{E}[X])\leq\frac{1}{4}$ necesitamos imponer la siguiente restricción sobre k:

$$\frac{4\cdot(2^n-\#\text{DNF-SAT}(\varphi))}{\varepsilon^2\cdot\#\text{DNF-SAT}(\varphi)} \quad\leq\quad k$$

Un FPRAS para #DNF-SAT: primer intento

Tomando entonces:

$$k \quad = \quad \left\lceil \frac{4 \cdot (2^n - \#\text{DNF-SAT}(\varphi))}{\varepsilon^2 \cdot \#\text{DNF-SAT}(\varphi)} \right\rceil$$

obtenemos la cota superior deseada para la probabilidad de error.

¡Pero el valor de k puede ser exponencial en n, lo cual significa que el algoritmo aleatorizado resultante es de tiempo exponencial!

- Por ejemplo, si #DNF-SAT(φ) es polinomial en n

Vamos a estudiar un segundo enfoque que sí da el resultado deseado.

- El primer enfoque no utilizaba el hecho de que φ está en DNF