Técnicas Fundamentales

Algoritmos codiciosos

Segundo semestre 2022

IIC2283

Prof. Nicolás Van Sint Jan

Outline

Algoritmos codiciosos: Codificaciones

Algoritmos codiciosos

Los **algoritmos codiciosos** son usados, en general, para resolver **problemas de optimización**.

El principio fundamental de este tipo de algoritmos es lograr un **óptimo global** tomando **decisiones locales que son óptimas**.

Existen dos componentes fundamentales de un algoritmo codicioso:

1. Una **función objetivo** a minimizar o maximizar.
2. Una **función de selección** usada para tomar decisiones locales óptimas.

¿Siempre se puede usar un **algoritmo codicioso** para resolver un problema de optimización?

Algoritmos codiciosos

Lamentablemente en muchos casos los algoritmos codiciosos **NO** encuentran un óptimo global.

- Sin embargo, en muchos casos pueden ser usados como **heurísticas** para encontrar valores de la función objetivo cercanos al óptimo.

Una segunda dificultad con los algoritmos codiciosos es que, en general, se necesita de una demostración formal para asegurar que encuentran el óptimo global.

- Aunque pueden ser algoritmos simples, en algunos casos no es obvio por qué encuentran el óptimo global.

Vamos a ver un ejemplo clásico de algoritmos codiciosos.

Almacenamiento de datos

Sea Σ un alfabeto. Dada una palabra $w \in \Sigma^*$ suponga que queremos **almacenar** w **utilizando los símbolos 0 y 1**.

- Esta palabra w puede ser pensada como un **documento** si Σ incluye las letras del alfabeto español, símbolos de puntuación y el espacio, por lo tanto puede ser muy larga.
- El uso de 0 y 1 corresponde a la idea de almacenar el texto en un computador.

Definimos entonces una función $\tau : \Sigma \to \{0,1\}^*$ que asigna a cada símbolo en $a \in \Sigma$ una palabra en $\tau(a) \in \{0,1\}^*$ con $\tau(a) \neq \varepsilon$

- Vamos a almacenar w reemplazando cada símbolo $a \in \Sigma$ que aparece en w por $\tau(a)$.
- Llamamos a τ una **Σ-codificación**.

Almacenamiento de datos

La **extensión** $\hat{\tau}$ de una Σ-codificación τ a todas las palabras $w \in \Sigma^*$ se define como:

$$\hat{\tau}(w) = \begin{cases} \varepsilon & w = \varepsilon \\ \tau(a_1) \cdots \tau(a_n) & w = a_1 \cdots a_n \text{ con } n \geq 1 \end{cases}$$

Vamos a almacenar w como $\hat{\tau}(w)$

- Si la Σ-codificación τ está fija (como el código ASCII) entonces no es necesario almacenarla
- Si τ no está fija entonces debemos almacenarla junto con $\hat{\tau}(w)$
  - En general $|w|$ es mucho más grande que $|\Sigma|$, por lo que el costo de almacenar τ es despreciable

Almacenamiento de datos

La función $\hat{\tau}$ debe especificar una traducción no ambigua:

$$\forall w_1, w_2 \in \Sigma^*. w_1 \neq w_2 \Rightarrow \hat{\tau}(w_1) \neq \hat{\tau}(w_2)$$

De esta forma podemos reconstruir el texto original dada su traducción. **(¿Por qué?)**.

Vale decir, $\hat{\tau}$ debe ser una **función inyectiva**.

¿Cómo podemos lograr esta propiedad?

Codificaciones de largo fijo

Es claro que para lograr una traducción no ambigua necesitamos que $\tau(a) \neq \tau(b)$ para cada $a, b \in \Sigma$ tal que $a \neq b$

Para obtener la misma propiedad para $\hat{\tau}$ podemos imponer la siguiente condición:

$$\forall a, b \in \Sigma.\ |\tau(a)| = |\tau(b)|$$

Si se cumple esta condición entonces diremos que τ es una **Σ-codificación de largo fijo**.

Ejercicio

Muestre que para tener una Σ-codificación de largo fijo basta con asignar a cada $a \in \Sigma$ una palabra $\tau(a) \in \{0,1\}^*$ tal que $|\tau(a)| = \lceil \log_2(|\Sigma|) \rceil$

Codificaciones de largo variable

Por otro lado, decimos que τ es una **Σ-codificación de largo variable** si

$$a, b \in \Sigma.\ |\tau(a)| \neq |\tau(b)|$$

¿Por qué nos conviene utilizar **codificaciones de largo variable**?

R: Podemos obtener representaciones más cortas para la palabra que queremos almacenar

Codificaciones de largo variable

Ejemplo

Suponga que $w = aabaacaab$.

- Para $\tau_1(a) = 00$, $\tau_1(b) = 01$ y $\tau_1(c) = 10$ tenemos que:

$$\hat{\tau}_1(w) \quad = \quad 000001000010000001$$

- Para $\tau_2(a) = 0$, $\tau_2(b) = 10$ y $\tau_2(c) = 11$ tenemos que:

$$\hat{\tau}_2(w) \quad = \quad 001000110010$$

Por lo tanto $|\hat{\tau}_2(w)| = 12 < 18 = |\hat{\tau}_1(w)|$

¿Por qué $\hat{\tau}_2$ es **inyectiva**?

Codificaciones de largo variable

Lema

Si existen $w_1, w_2 \in \Sigma^*$ tales que $w_1 \neq w_2$ y $\hat{\tau}(w_1) = \hat{\tau}(w_2)$, entonces existen $a, b \in \Sigma$ tales que $a \neq b$ y $\tau(a)$ es un prefijo de $\tau(b)$

Demostración

Suponga que $w_1 \neq w_2$, $\hat{\tau}(w_1) = \hat{\tau}(w_2)$ y

$$\begin{array}{lcll} w_1 & = & a_1 \dots a_m & m \geq 1 \\ w_2 & = & b_1 \dots b_n & n \geq 1 \end{array}$$

Además, sin perdida de generalidad suponga que $m \leq n$.

Si w_1 es **prefijo propio** de w_2 entonces $\hat{\tau}(w_1)$ es prefijo propio de $\hat{\tau}(w_2)$

- Puesto que $\tau(a) \neq \varepsilon$ para cada $a \in \Sigma$.

Dado que $\hat{\tau}(w_1) = \hat{\tau}(w_2)$, tenemos entonces que w_1 no es prefijo propio de w_2.

Codificaciones de largo variable

Demostración

Sea $k = \min_{1 \leq i \leq m} a_i \neq b_i$

- k está bien definido puesto que $w_1 \neq w_2$ y w_1 no es prefijo propio de w_2

Dado que $\hat{\tau}(a_1 \cdots a_{k-1}) = \hat{\tau}(b_1 \cdots b_{k-1})$ y $\hat{\tau}(w_1) = \hat{\tau}(w_2)$, concluimos que $\hat{\tau}(a_k \cdots a_m) = \hat{\tau}(b_k \cdots b_n)$

Tenemos entonces que $\tau(a_k) \cdots \tau(a_m) = \tau(b_k) \cdots \tau(b_n)$, de lo cual se deduce que $\tau(a_k)$ es un prefijo de $\tau(b_k)$ o $\tau(b_k)$ es un prefijo de $\tau(a_k)$

- Lo cual concluye la demostración puesto que $a_k \neq b_k$ □

Codificaciones libres de prefijos

Decimos que una Σ-codificación τ es **libre de prefijos** si para cada $a, b \in \Sigma$ tales que $a \neq b$ se tiene que $\tau(a)$ no es un prefijo de $\tau(b)$

Ejemplo

Para $\Sigma = \{a, b, c\}$ y $\tau(a) = 0$, $\tau(b) = 10$, $\tau(c) = 11$, se tiene que τ es libre de prefijos.

Corolario

Si τ es una codificación libre de prefijos, entonces $\hat{\tau}$ es una **función inyectiva**.

Frecuencias relativas de los símbolos

Palabra a almacenar: $w \in \Sigma^*$

Para $a \in \Sigma$ definimos $\mathrm{fr}_w(a)$ como la **frecuencia relativa** de a en w, vale decir, el número de apariciones de a en w dividido por el largo de w

- Por ejemplo, si $w = aabaacaab$, entonces $\mathrm{fr}_w(a) = \frac{2}{3}$ y $\mathrm{fr}_w(c) = \frac{1}{9}$

Para una Σ-codificación τ definimos el **largo promedio** para w como:

$$\mathrm{lp}_w(\tau) \quad = \quad \sum_{a \in \Sigma} \mathrm{fr}_w(a) \cdot |\tau(a)|$$

Tenemos entonces que $|\hat{\tau}(w)| = \mathrm{lp}_w(\tau) \cdot |w|$

- Por lo tanto queremos una Σ-codificación τ que **minimice** $\mathrm{lp}_w(\tau)$

Frecuencias relativas de los símbolos y la función objetivo

Problema de optimización a resolver

Dado $w \in \Sigma^*$, encontrar una Σ-codificación τ libre de prefijos que minimice el valor $\mathrm{lp}_w(\tau)$

La función objetivo del algoritmo codicioso es entonces $\mathrm{lp}_w(x)$

- Queremos minimizar el valor de esta función

Frecuencias relativas de los símbolos y la función objetivo

La función $\mathrm{lp}_w(x)$ se define a partir de la función $\mathrm{fr}_w(y)$

- No se necesita más información sobre w. En particular, no se necesita saber cuál es el símbolo de w en una posición específica

Podemos entonces trabajar con funciones de frecuencias relativas en lugar de palabras

- La entrada del problema no va a ser w sino que fr_w

Frecuencias relativas de los símbolos y la función objetivo

Decimos que $f : \Sigma \to (0, 1)$ es una función de frecuencias relativas para Σ si se cumple que $\sum_{a \in \Sigma} f(a) = 1$

Dada una función f de frecuencias relativas para Σ y una Σ-codificación τ, el largo promedio de τ para f se define como:

$$\mathrm{lp}_f(\tau) \quad = \quad \sum_{a \in \Sigma} f(a) \cdot |\tau(a)|$$

La entrada del problema es entonces una función f de frecuencias relativas para Σ, y la función objetivo a minimizar es $\mathrm{lp}_f(x)$