Solucionando el problema de la mochila en el Modelo Sticker

paper por Mario A. Jimenez

Ernesto Mancebo

Universidad de Sevilla

Febrero 2020

Contenido

- Modelo propuesto por Sam Roweis.
- Inspirado en Cadenas de ADN.
- Basado en operaciones de filtrado.
- Distinguido por utilizar Memoria de Acceso Aleatorio y la manera de representar la data.

Las cadenas de memoria (también llamadas moléculas σ) son cadenas de forma (n, k, m), tal que $n \geq k.m$, siendo n la longitud de la cadena, k la cantidad de sub-cadenas , y m la longitud de cada sub-cadena p.
Cada región m tiene un estado *"apagado"/"encendido"* o *"0"/"1"*

Un Tubo (T) consiste en un multiconjunto de complejos de memoria o de moléculas σ del mismo tipo.

Se puede ilustrar de la manera: $\begin{bmatrix} \sigma_0 \\ \sigma_i \\ \sigma_{i+1} \end{bmatrix}$

- $mezclar(T_1, T_2)$
- $separar(T, i)$
- $encender(T, i)$
- $apagar(T, i)$
- $leer(T)$

- Complejo de memoria de forma (k, l) tal que $(1 \leq k \leq l)$.
- Se inicializan las primeras $k - l$ regiones *"encendidas"/"apagadas"* en todas sus posibles combinaciones.

Subrutinas

Partiendo de los siguientes elementos:

- $A = \{1, \cdots, p\}$
- $B = \{b_1, \cdots, b_s\} \subseteq A$
- $F = \{D_1, \cdots, D_t\} \subseteq P(A)$

Se busca ordenar F con respecto a su cardinalidad en B, o en otras palabras, la cantidad de elementos que conincidan de manera $B \cap D_i$.

Codificando los subconjuntos F

Sea σ una molécula para el tubo T_0, la misma se codifica de forma $T_0 = \{\{\sigma : |\sigma| = p \wedge \exists j(\chi_{D_j} = \sigma)\}\}$; tal que χ_{D_j} es la función característica en A, siendo $(\chi_{D_j}(i) = 1$ si $i \in D_j$ de lo contrario $\chi_{D_j} = 0$ si $i \in A - D_j)$

Resultado

- Una vez concluido el paso i, se tendrán $i+1$ tubos, empezando en T_0.
- Se tendrá $\forall\sigma(\sigma \in T_j \rightarrow |\sigma \in \{b_1, \cdots, b_i\}| = j)$.

Algoritmo

procedure $Cardinal\_Sort(T_0, B)$
  for $i = 1$ to s **do**
    $(T_0, T_0') = separar(T_0, b_i)$
    for $j = 0$ to $i - 1$ **do**
      $(T_{j+1}', T_j'') = separar(T_j, b_i)$
      $T_j = mezclar(T_j', T_j'')$
    end for
    $T_i = T_i'$
  end for
  Return $[T_0, ..., T_s]$
end procedure

Traza

A fin de ilustrar el comportamiento:

- $A : \{0, 1, 2, 3, 4, 5, 6\}$
- $B : \{1, 2, 4\}$
- $F : \{[2, 6], [3], [4], [2, 4]\}$

Los elementos que cumplen $B \cap D_j$ en F son: $\{[2, 6], [3], [4], [2, 4]\}$.

F codificado y procesado

Codificando F para llevarlo a un tubo tendremos:

$$\begin{bmatrix} [0,0,1,0,0,0,1] \\ [0,0,0,1,0,0,0] \\ [0,0,0,0,1,0,0] \\ [0,0,1,0,1,0,0] \end{bmatrix}$$

Tras la ejecución de *Cardinal_Sort* tendremos:

$$\begin{bmatrix} T_0 : [[0,0,0,1,0,0,0]] \\ T_1 : [[0,0,0,0,1,0,0],[0,0,1,0,0,0,1]] \\ T_2 : [[0,0,1,0,1,0,0]] \\ T_3 : [] \\ T_4 : [] \\ T_5 : [] \\ T_6 : [] \end{bmatrix}$$

Otras Notaciones

- $Cardinal\_Sort(T_0)$ cuando $B = A$.
- $Cardinal\_Sort(T_0, l, k)$ cuando $B = \{l, l+1, \cdots, k\}$.

El propósito de la subrutina de llenado es manipular el tubo T_0 las moléculas en $(\sigma(i))$ y codifiquen su valor con respecto a f en el sub-conjunto A, ambos se definen como:

- $A = \{1, \cdots, p\}$
- $r \in \mathbb{N}$
- $f : A \to \mathbb{N}$

Delimitación de Regiones

Podemos considerar la idea de segmentar cada molécula σ en T_0 en posibles regiones como:

- $(A\sigma) = \sigma(1) \cdots \sigma(p)$
- $(L\sigma) = \sigma(p+1) \cdots \sigma(p+r)$
- $(F\sigma) = \sigma(p+r+1) \cdots \sigma(p+r+q_f)$
- $(R\sigma) = \sigma(p+r+q_f+1) \cdots$

Para este apartado denotamos que si $B \subseteq A$, decimos que $f(B) = \sum_{i \in B} f(i)$. Tal suma nos indica el espacio a reservar en el complejo de memoria T_0 para codificar $f(B)$.
Asimismo, definimos $q_f = f(A)$, tal que $A_i = \{0, \cdots, i\}$ siendo $(i \leq i \leq p)$.
Por último, tenemos que r corresponde a la región comprendida entre p y $f(B)$.
Todos estos ingredientes nos dan soporte para definir el tubo T_0 de moléculas σ cuyo $k \geq p + r + q_f$.

Algoritmo

procedure *Parallel_Fill*(T_0, f, p, r)
  for $i = 1$ to p **do**
    $(T^+_{i,0}, T^-_i) = separar(T_{i-1}, i)$
    for $j = 1$ to $f(i)$ **do**
      $T^+_{i,j} = encender(T^+_{i,j}, p + r + f(A_{i-1}) + j)$
    end for
    $T_i = mezclar(T^+_{i,f(i)}, T^-_i)$
  end for
  Return T_0
end procedure

Traza

A fin de ilustrar el comportamiento, estudiemos lo siguiente:

- $A : \{1, 2, 3, 4\}$
- $B : \{2, 3, 4\}$
- $T_0 : \{[2], [4], [3]\}$

Para el tubo T_0 lo codificaríamos de la siguiente forma:

$$\begin{bmatrix} [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \\ [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \\ [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \end{bmatrix}$$

T_0 procesado

Una vez el tubo T_0 sea procesado por *Parallel_Fill*, tendremos:

$$\begin{bmatrix} [0,1,0,0,1,1,1,0,0,0,0,0,0,0] \\ [0,0,0,1,1,1,1,1,1,1,1,1,1,1] \\ [0,0,1,0,1,1,1,1,1,1,0,0,0,0] \end{bmatrix}$$

Se busca determinar si existe en B un subconjunto cuyo valor equivalga a k.
En ese sentido, definimos $A = \{1, \cdots, p\}, k \in \mathbb{N}, w : A \to \mathbb{N}$, siendo w la función peso tal que $k \leq w(A) = q_w$.

Algoritmo

```
procedure Subset_Sum(p, w, k)
    q_w = ∑_{i=1}^{p} w(i)
    T_0 = Libería(p + q_w, p)
    T_1 = Parallel_Fill(T_0, w, p, 0)
    T_k = Cardinal_Sort(T_1, p + 1, p + q_w)[k]
    leer(T_k)
end procedure
```

Traza

Teniendo como ejemplo el T_0 anterior:

$$\begin{bmatrix} [0,1,0,0,1,1,1,0,0,0,0,0,0,0] \\ [0,0,0,1,1,1,1,1,1,1,1,1,1,1] \\ [0,0,1,0,1,1,1,1,1,1,0,0,0,0] \end{bmatrix}$$

Para un $k = 10$, una vez el T_0 sea procesado por *Subset_Sum* tendríamos como salida: $[0,0,0,1,1,1,1,1,1,1,1,1,1,1]$