Modelos Problemas 1 al 4 Práctica 7 – Análisis y Diseño de Datos y Algoritmos

Alejandro Fernández Trigo

1. Alumnos

Una academia de inglés tiene n alumnos a ser repartidos en m grupos (n múltiplo de m). Cada grupo tiene distinto horario y profesor. De cada alumno se conoce la afinidad que tiene para pertenecer a cada uno de los grupos (valor entero en el rango [0,5]). Se desea conocer el reparto de alumnos en grupos, de forma que todos los grupos deben tener el mismo número de alumnos, maximizando la afinidad total conseguida para todos los alumnos, y teniendo en cuenta que no está permitido asignar un alumno a un grupo para el que presente afinidad 0.

$$
\begin{aligned}
&max\ \Sigma_{i=0}^{n-1} af(i, x_i) \\
&C_{i=0|x_i=j a(i,x_i)>0}^{n-1} x_i = \frac{n}{m}, \qquad j \in [0,m) \\
&0 \le x_i < m, \qquad i \in [0,n) \\
&int\ \ x_i\,, \qquad i \in [0,n)
\end{aligned}
$$

Academia*Vertex:*

Propiedades:

- *index: Integer, básica*
- *pl: Integer[m], básica, plazas libres por grupos*
- *n: Integer, compartida, número de alumnos*
- *m: Integer, compartida, número de grupos*
- *Neg: Integer, compartida, número estudiantes por grupo*

Interpretación:

Encontrar la asignación de estudiantes a grupos, desde index hasta el final, que optimicen la afinidad.

Igualdad

- *Dos problemas son iguales si lo son index, pl*

Es válido

- *(index>=0,index<=n) y (pl[k]>=0, para k>=0 y k<m)*

Factoría:

- *inicial(): Crea el problema (0,[Neg,Neg,Neg...,Neg])*
- *goal(v) = p.i ==n*

Casos base:

1. *p.i == n-1*

Solución caso Base

1. *Tiene solución*

Acciones:

- $A_i = \{a{:}0..m-1 | pl[a] > 0 \,\&\&\, af(index,a) > 0\}$

Vecino

Caso general: $neighbord(a) = (i+1, pl'),\ \ pl'[a] = pl[a] - 1$

Peso de la arista

Peso: $w = af(index, a)$

Peso del camino:

Suma de los pesos de las aristas

Solución Voraz

- Acción Voraz: $\underset{j:0..m-1}{\text{argmax}}(af(index,j)\,|pl[j] > 0)$

Heurística:

- $\sum_{i=index}^{n-1} \max_{j:0..m-1}(af(i,j)\,|pl[j] > 0)$

Solución:

- *List<Integer>* que indicarían el número del grupo al que se asigna cada alumno

2. Abogados

Un bufete de abogados cuenta con un equipo de n personas que deben analizar m casos relacionados entre sí (m>=n), y deben terminar dicho análisis global lo antes posible para lo que trabajarán en paralelo. Cada caso será analizado por un único abogado, y cada abogado puede analizar varios casos. Se conoce el tiempo (en horas) que se estima que tarda cada abogado en analizar cada caso concreto (dicho tiempo puede diferir para cada caso en función de qué abogado realice el análisis). Determine cuál es la mejor asignación de casos a abogados para conseguir el objetivo indicado (terminar de analizar todos los casos lo antes posible).

$$\min \max_{i:0..n-1} \sum_{j=0|x_j=i}^{m-1} c(i,j)$$
$$0 \le x_j < n, \quad j \in [0, m-1]$$
$$int\ x_j, \quad j \in [0, m-1]$$

BufeteVertex:

Propiedades:

- *index: Integer, básica*
- *ca: Integer[n], básica, carga de abogados*
- *cMax: Integer, derivada, carga del abogado más cargado*
- *cMin: Integer, derivada, carga del abogado menos cargado*
- *aMax: Integer, derivada, abogado más cargado*
- *aMin, Integer, derivada, abogado menos cargado*
- *n: Integer, compartida, número de abogados*
- *m: Integer, compartida, número de casos*

Interpretación:

Encontrar la asignación de casos a abogados, desde index hasta el final, que minimicen cMax, teniendo en cuenta las cargas ya acumuladas para los abogados

Igualdad

- *Dos problemas son iguales si lo son index, ca*

Hay correcciones al modelo de Abogados en la última página.

Es válido

- *index>=0, index<=m*

Factoría:

- *inicial(): Crea el problema (0,[0,0,0...,0])*
- *goal(v) = p.i == m*

Casos base:

1. *p.i == m-1*

Solución caso Base

1. *Tiene solución*

Acciones:

- $A_i = \{aMin\}$, si p.i = m-1
- $A_i = \{a: 0..n-1\}$,

Vecino

Caso general: $neighbor(a) = (i+1, ca'),\ \ ca'[a] = ca[a] + c(a,i)$

Peso de la arista

No

Peso del camino:

El peso del último vértice

Solución Voraz

- Acción Voraz: *aMin, (y ordenados los casos de mayor a menor duración)*

Heurística:

- *0*

Solución:

Map<Integer,Integer> que indicarían el abogado al que se asigna cada caso. En la solución habría que incluir las propiedades derivadas adecuadas

3. Productos

Se tienen n productos, cada uno de los cuales tiene un precio y presenta una serie de funcionalidades (el mismo producto puede tener más de una funcionalidad). Se desea diseñar un lote con una selección de dichos productos que cubran un conjunto de funcionalidades deseadas entre todos productos seleccionados al menor precio.

$$\min \sum_{i=0}^{n-1} p_i\, x_i$$
$$\bigcup_{i=0|x_i=1}^{n-1} f_i \supset D$$
$$bin\, x_i, \quad i \in [0, n-1]$$

Productos*Vertex:*

Propiedades:

- *index: Integer, básica*
- *fc: Set<Integer>, básica, funcionalidades por cubrir*
- *n: Integer, compartida, número de productos*
- *D: Set<Integer>, compartida, funcionalidades deseadas*

Interpretación:

Encontrar la elección adecuada de productos, desde index hasta el final, que minimicen el precio total y cubran todas las funcionalidades deseadas

Igualdad

- *Dos problemas son iguales si lo son index, fc*

Es válido

- *index>=0,index<=n, los elementos de fc en D*

Factoría:

- *inicial(): Crea el problema (0,D)*
- *goal(v) = p.i ==n*

Peso de la arista

$a*p_i$

Peso del camino:

La suma de los pesos de las aristas

Solución Voraz

- Acción Voraz: *max(a: A_i), estando los productos ordenados de mayor a menor (número de funcionalidades ofrecidas)/precio*

Heurística:

- fc = Ø?0:min (p_i, index <= i < n)

Solución:

- *Set<Integer>* que indicarían los productos elegidos. En la solución habría que incluir las propiedades derivadas adecuadas

Casos base:

1. *p.index == n-1*
2. *fc = Ø*

Solución caso Base

1. *Tiene solución si (fc = Ø) ó ($fc - f_{n-1} = \emptyset$)*
2. *Tiene solución*

Acciones:

- A_i ={}, si $i = n$
- A_i ={0}, si $fc = \emptyset$
- A_i ={1}, si (i = n-1) $y\ (fc - f_{n-1} = \emptyset)$
- A_i ={}, si (i = n-1) $y\ (fc - f_{n-1} \neq \emptyset)$
- A_i ={0,1} en otro caso

Vecino

Para el caso base fc = Ø

- $neighbor(0) = (n,\ \emptyset)$

Caso general:

- $neighbor(0) = (i+1,\ fc)$
- $neighbor(1) = (i+1, fc - f_{index})$

4. Conjuntos

Dado un conjunto de enteros determinar si puede particionarse en tres subconjuntos de manera que la suma de elementos en los tres subconjuntos sea la misma, y que el tamaño de uno de ellos sea lo menor posible.

$$\min C_{i=0|x_i=0}^{n-1} x_i$$
$$x_i \leq 2, \quad i \in [0, n-1]$$
$$\sum_{i=0|x_i=j}^{n-1} e_i = N, \quad j \in \{0,1,2\}$$
$$int\ x_i, \quad i \in [0, n-1]$$

Particion*Vertex:*

Propiedades:

- *index: Integer, básica*
- *vr: Integer[3], básica, cada elemento es N menos la suma de los elementos asignados a cada conjunto*
- *n: Integer, compartida, número de elementos*

Interpretación:

Encontrar a qué subconjunto pertenece cada elemento, desde index hasta el final, que minimice el número de elementos de uno de los subconjuntos

Igualdad

- *Dos problemas son iguales si lo son index, vr*

Es válido

- *index>=0,index<=n, vr[i]>=0, i en 0..2*

Factoría:

- *inicial(): Crea el problema (0,[N,N,N])*
- *goal(v) = p.i ==n*

Casos base:

1. *p.i == n-1*

Solución caso Base

1. *Tiene solución si existe a tal que* $vr[i] = 0, i \neq a, vr[a] - e_i = 0$

Acciones:

- En el caso p.i = n-1
    - {a} Si $vr[i] = 0, i \neq a, vr[a] - e_i = 0$
    - {} En otro caso
- $A_i = \{a: 0..2 | vr[a] - e_i \geq 0\}$, caso general

Vecino

Caso general: $neighbor(a) = (i+1,\ vr'), vr'[a] = \ vr[a] - e_i$

Peso de la arista

a==0?1:0

Peso del camino:

La suma de los pesos de las aristas

Solución Voraz

- Acción Voraz*:* $\underset{j:A_i}{\operatorname{argmax}}\, vr[a]$

Heurística:

- *vr[0]==0?0:1*

Solución:

Map<Integer,Integer> que indicarían los conjuntos asignados a cada elemento. En la solución habría que incluir las propiedades derivadas adecuadas

5. Correcciones al modelo de Abogados:

Donde j es casos e i es abogados.

Casos base:

1. $p.j == m-1$

Solución caso Base

1. *Tiene solución* $\underset{0 \le i < n}{\operatorname{argmin}} \left(carga[i] + c(i,j)\right)$

Acciones:

- $A_j = \{\cancel{aMin}\}$, si $p.j = m-1$
- $A_j = \{a: 0..n-1\}$, tal que no existe $a' < a$ con $carga[a'] = carga[a]$

Vecino

Caso general: $neighbor(a) = (j+1, ca')$, $ca'[a] = ca[a] + c(a,j)$