Prueba de aula Algorítmica (11593)

10 de enero de 2023

NOMBRE:

1 — 2+3=5 puntos

Dado un tablero de ajedrez de tamaño $N \times N$ donde cada casilla $(fila, columna)$ tiene asignado un valor, haz una traza de un algoritmo Ramificación y Poda que sitúe N torres en dicho tablero de modo que las torres no se amenacen entre sí y que la suma de las casillas ocupadas por las torres sea máxima. (Nota: las torres se mueven libremente en horizontal y vertical.) Haz la traza para esta instancia con $N = 4$.

		columnas 1	2	3	4
filas	1	6	4	1	2
	2	7	1	1	4
	3	3	2	4	6
	4	5	2	4	2

a) Sigue una estrategia por primero el mejor (en caso de empate, el estado más cercano a una solución) y con poda implícita.

b) Debes usar el esquema que inicializa la variable mejor solución $\hat{x}$ a una solución factible usando un algoritmo voraz. Explica el algoritmo, calcula su coste temporal y calcula qué solución tendrías para la instancia dada. Tu algorimo, ¿devuelve siempre la solución óptima? **(2 puntos)**

c) Para la traza sobre la instancia presentada, ten en cuenta lo siguiente: La traza debe mostrar el *conjunto de estados activos* de cada iteración del algoritmo indicando la cota optimista de cada estado (ej: como superíndice) y subrayando el estado que se selecciona para explorar en la siguiente iteración. Hay que indicar también si se actualiza la variable mejor solución y la poda implícita u otras podas. **(3 puntos)**

Solución:

Algoritmo voraz: Podemos inicializar la variable mejor solución con una solución voraz: elegir la columna de valor máximo para la torre de la primera fila; el máximo para la torre de la segunda fila entre las casillas libres; y así sucesivamente se llegaría a la solución $(1, 4, 3, 2)$ con un valor asociado de 6+4+4+2=16. Este algoritmo tiene un coste $O(N^2)$ y es subóptimo. Esto se demuestra porque, por ejemplo, para esta instancia, ya devuelve una solución subóptima, pues la solución óptima veremos que es mejor que esta.

columnas

		1	2	3	4	
filas	1	**6**	4	1	2	torre fila 1 se coloca en columna 1
	2	7	1	1	4	
	3	3	2	4	6	
	4	5	2	4	2	

columnas

		1	2	3	4	
filas	1	**6**	4	1	2	torre fila 1 se coloca en columna 1
	2	-	1	1	**4**	torre fila 2 se coloca en columna 4
	3	-	2	4	6	
	4	-	2	4	2	

columnas

		1	2	3	4	
filas	1	**6**	4	1	2	torre fila 1 se coloca en columna 1
	2	-	1	1	**4**	torre fila 2 se coloca en columna 4
	3	-	2	**4**	-	torre fila 3 se coloca en columna 3
	4	-	2	4	-	

		columnas				
		1	*2*	*3*	*4*	
filas	*1*	**6**	4	1	2	torre fila 1 se coloca en columna 1
	2	-	1	1	**4**	torre fila 2 se coloca en columna 4
	3	-	2	-	-	torre fila 3 se coloca en columna 3
	4	-	**2**	-	-	torre fila 3 se coloca en columna 2

Algoritmo RyP: La solución vendrá dada por una configuración válida de las torres en el tablero, que se puede representar como una tupla de 4 elementos (x_1, x_2, x_3, x_4) donde x_i indica la columna donde se coloca la torre de la fila i. La restricción es que no puede haber dos torres en la misma columna (en la misma fila está implícito por la representación elegida), por lo que, en realidad, las soluciones factibles serán el conjunto de permutaciones de 1 a N.

Una posible cota superior para este problema vendría dada por el máximo valor de las columnas en cada fila y asumir que las torres restantes se colocan ahí, sin tener en cuenta si la columna que tiene esa puntuación ya está ocupada. Estos valores se pueden preprocesar con un coste $O(N^2)$ y tienen un valor para la instancia: $maxcol = (6, 7, 6, 5)$. Si el cálculo de la cota se realiza de forma incremental, el coste será constante:

$$F((x_1, x_2, \ldots, x_k, ?)) = F((x_1, x_2, \ldots, x_{k-1}, ?)) + v(k, x_k) - maxcol_k$$

Inicializamos el conjunto de estados activos con el estado que representa todo el espacio de búsqueda (ninguna torre posicionada), con un valor de cota igual a $6 + 7 + 6 + 5 = 24$. En cada iteración seleccionaremos como siguiente a ramificar el más prometedor de acuerdo a su valor de cota. La variable mejor solución está inicializada con la solución $(1, 4, 3, 2)$ con un valor asociado de $6 + 4 + 4 + 2 = 16$. Aunque en el algoritmo de ramificación y poda el conjunto de estados activos se representaría con un maxheap, en la traza lo representaremos como un conjunto sin ordenar.

$$A^0 = \{\overbrace{\underline{(?)}}^{24}\}$$

Al seleccionar el estado (?) se obtienen 4 nuevos estados:

$$A^1 = \{\overbrace{\underline{(1,?)}}^{24-6+6=24}, \overbrace{(2,?)}^{24-6+4=22}, \overbrace{(3,?)}^{24-6+1=19}, \overbrace{(4,?)}^{24-6+2=20}\}$$

Seleccionamos para ramificar el estado $(1, ?)$:

$$A^2 = \{\overbrace{(1,2,?)}^{24-7+1=18}, \overbrace{(1,3,?)}^{24-7+1=18}, \overbrace{(1,4,?)}^{24-7+4=21}, \overbrace{\underline{(2,?)}}^{22}, \overbrace{(3,?)}^{19}, \overbrace{(4,?)}^{20}\}$$

Seleccionamos para ramificar el estado $(2, ?)$. Uno de sus hijos, el estado $(2, 3, ?)$, no se guarda pues su cota optimista $(22 - 7 + 1 = 16)$, no puede mejorar la colución que ya tenemos:

$$A^3 = \{\overbrace{(1,2,?)}^{18}, \overbrace{(1,3,?)}^{18}, \overbrace{(1,4,?)}^{21}, \overbrace{\underline{(2,1,?)}}^{22-7+7=22}, \overbrace{(2,4,?)}^{22-7+4=19}, \overbrace{(3,?)}^{19}, \overbrace{(4,?)}^{20}\}$$

Seleccionamos para ramificar el estado $(2, 1, ?)$:

$$A^4 = \{\overbrace{(1,2,?)}^{18}, \overbrace{(1,3,?)}^{18}, \overbrace{(1,4,?)}^{21}, \overbrace{(2,1,3,?)}^{22-6+4=20}, \overbrace{\underline{(2,1,4,?)}}^{22-6+6=22}, \overbrace{(2,4,?)}^{19}, \overbrace{(3,?)}^{19}, \overbrace{(4,?)}^{20}\}$$

Seleccionamos para ramificar el estado $(2, 1, 4, ?)$ y obtenemos la solución $(2, 1, 4, 3)$, de valor 21, que mejora la solución actual.

$$A^5 = \{\overbrace{(1,2,?)}^{18}, \overbrace{(1,3,?)}^{18}, \overbrace{\underline{(1,4,?)}}^{21}, \overbrace{(2,1,3,?)}^{20}, \overbrace{(2,4,?)}^{19}, \overbrace{(3,?)}^{19}, \overbrace{(4,?)}^{20}\}$$

Seleccionamos el estado $(1, 4, ?)$ para explorar, pero este ya no puede mejorar la solución que tenemos. Por tanto, como este estado es el mejor de todos los que restan, el algoritmo termina y devuelve como mejor solución $(2, 1, 4, 3)$, de valor 21.

La empresa de transporte Paraná desea fletar un avión para transportar su mercancía, formada por N paquetes, de pesos $p_1, p_2, \ldots, p_N$. Cada paquete, adicionalmente, viene marcado con su beneficio $b_1, b_2, \ldots, b_N$. Hay que cargar los paquetes distribuidos entre las 2 bodegas, de capacidades A y B. Para asegurar la estabilidad del avión, la diferencia de los pesos cargados en cada bodega no puede superar un umbral Θ. Sabiendo que no se pueden transportar todos los paquetes, pues el peso total de estos, $\sum_1^N p_i$, supera la capacidad de carga del avión, $A + B$, se desea seleccionar el subconjunto de paquetes que maximice el beneficio total, que viene dado por la suma del beneficio de los paquetes cargados en el avión. Para resolver el problema por Ramificación y poda, se pide:

a) Expresa el problema en términos de optimización: expresa formalmente el conjunto de soluciones factibles, la función objetivo a maximizar y la solución óptima. Explica brevemente cómo expresas una solución x del conjunto X.

b) Describe los siguientes conceptos sobre los estados que serán necesarios para el algoritmo:

   1) Representación de un estado (no terminal).
   2) Condición para que un estado sea solución.
   3) Identifica el estado inicial que representa todo el conjunto de soluciones factibles.

c) Define una función de ramificación. Contesta a las siguientes cuestiones:

   1) Explica la función.
   2) Define la función (en python o en lenguaje matemático).

d) Diseña una cota optimista. Contesta a las siguientes cuestiones:

   1) Explica la cota (caso general, de un estado intermedio).
   2) Explica el cálculo de la cota del estado inicial.
   3) Define la cota (en python o en lenguaje matemático). Calcula el coste temporal.
   4) Estudia si se puede mejorar el cálculo de la cota, haciéndolo incremental. Define la cota así definida (en python o en lenguaje matemático). Calcula el coste temporal.

Solución: Este problema es una pequeña variante del problema de las múltiples mochilas, con 2 mochilas (bodegas), teniendo en cuenta adicionalmente que la diferencia de peso entre las dos bodegas no puede superar el umbral dado. Podemos representar una solución como una tupla de N elementos $(x_1, x_2, ..., x_N)$, donde $x_i = 0$ indica que el paquete i-ésimo no se carga en el avión, $x_i = 1$ indica que el paquete i-ésimo se carga en la bodega A y con $x_i = 2$ indicamos que se carga en la bodega B. Por ejemplo, si hay 6 paquetes, una solución sería $(0, 1, 1, 2, 0, 1)$, esto es, los paquetes segundo, tercero y sexto se cargan en la bodega A, el paquete cuarto en la bodega B, y los paquetes primero y quinto no se cargan. Sería una solución siempre que se cumplan las restricciones (caben los paquetes en cada bodega y no se supera la diferencia de peso Θ entre ambas). Formalizamos:

$$X = \left\{ (x_1, x_2, \ldots, x_N) \in \{0,1,2\}^N \mid \sum_{1 \le i \le N, x_i = 1} w_i \le A, \sum_{1 \le i \le N, x_i = 2} w_i \le B, \mid \sum_{1 \le i \le N, x_i = 1} w_i - \sum_{1 \le i \le N, x_i = 2} w_i \mid \le \Theta \right\}$$

$$f((x_1, x_2, \ldots, x_N)) = \sum_{1 \le i \le N, x_i \neq 0} b_i$$

$$\hat{x} = \arg\max_{x \in X} \sum_{1 \le i \le N, x_i \neq 0} b_i$$

Un estado se representará con una secuencia incompleta $(x_1, x_2, ..., x_k)$, con $k < N$. Tendrá un coste espacial de $O(N)$. El estado inicial se puede representar como una tupla vacía (?). Para que un estado sea solución la tupla deberá tener una talla igual a N y además que se cumplan la restricciones: la suma de los pesos de los paquetes cargados en la primera bodega no puede superar A, en la segunda bodega no puede superar B y la diferencia máxima de peso debe ser Θ. La función de ramificación dará lugar a tres nuevos estados como mucho: el estado que añade la decisión de no cargar el siguiente paquete $k+1$ y, hasta dos nuevos estados hijo si la suma del peso de los paquetes cargados hasta el momento en cada bodega más el del nuevo paquete de peso $p_k + 1$ no sobrepasa la capacidad de esa bodega:

$$\begin{aligned} branch(x_1, x_2, ..., x_k, ?) = \{&(x_1, x_2, ..., x_k, 0, ?), \\ &(x_1, x_2, ..., x_k, 1, ?) \text{ si } \sum_{1 \le i \le N, x_i = 1} p_i + p_{k+1} \le A, \\ &(x_1, x_2, ..., x_k, 2, ?) \text{ si } \sum_{1 \le i \le N, x_i = 2} p_i + p_{k+1} \le B\} \end{aligned}$$

Para que la comprobación de la condición de no sobrepasar la capacidad de cada bodega sea eficiente, habrá que mantener en el estado el peso acumulado hasta el momento o, equivalentemente, la capacidad libre, en cada bodega.

La cota para un estado $(x_1, \ldots, x_k, ?)$, con $k < N$, estará compuesta por la suma de la función objetivo aplicada a la parte conocida, $\sum_{1 \le i \le k; x_i \neq 0} b_i$, más una estimación optimista de lo que queda por completar.

En pricipio, la cota optimista "cabe todo", eliminando además la restricción de que la carga debe estar distribuida entre las dos bodegas de forma que no supere un umbral dado, es la trivial (es decir, añadir $\sum_{k+1 \le i \le N} b_i$ como estimación optimista de la parte desconocida). Esta cota se puede calcular de forma incremental

- En el estado inicial: $cota\_superior(?) = \sum_{1 \le i \le N} b_i$, con coste $O(N)$
- En un estado intermedio de forma incremental con coste $O(1)$:

$$cota\_superior(x_1, x_2, \ldots, x_k, 0, ?) = cota\_superior(x_1, x_2, \ldots, x_k, ?) - b_{k+1}$$

$$cota\_superior(x_1, x_2, \ldots, x_k, a, ?) = cota\_superior(x_1, x_2, \ldots, x_k, ?), \text{para } a \in [1, 2]$$

Para diseñar una cota superior más ajustada podemos utilizar, por ejemplo, el hecho de que las múltiples mochilas con fraccionamiento son equivalentes a una sola mochila de capacidad la suma de las capacidades. Por tanto, podríamos considerar como cota optimista para la parte desconocida el resultado de aplicar el algoritmo voraz de la mochila con fraccionamiento. En este caso, la capacidad libre de la "mochila" (las dos bodegas del avión) es $(A - \sum_{1 \le i \le k, x_i = 1} p_i) + (B - \sum_{1 \le i \le k, x_i = 2} p_i)$ en la que seleccionar paquetes a cargar desde $k+1$ hasta N. Para ello sería de utilidad un preproceso para ordenar los N paquetes según su beneficio unitario, con un coste$O(N \log N)$.