Departamento de Informática
Universidad Carlos III de Madrid

Grado en Informática

Heurística y Optimización

22 de Enero de 2014

Normas generales del examen

① El tiempo para realizar el examen es de **4 horas**

② No se responderá a ninguna pregunta sobre el examen transcurridos los primeros **30 minutos**

③ Cada pregunta debe responderse en páginas separadas en el mismo orden de sus apartados. Si no se responde, se debe entregar una página en blanco

④ Escribe con claridad y en limpio, de forma ordenada y concisa

⑤ Si se sale del aula, no se podrá volver a entrar durante el examen

⑥ No se puede presentar el examen escrito a lápiz

Pregunta 1 (2 puntos)

Un aeropuerto dispone de tres pistas de aterrizaje. En las dos primeras sólo pueden aterrizar aviones *pequeños* o *medianos*; la tercera, sin embargo, la pueden utilizar aviones *medianos* y *grandes*. El tamaño de cada avión afecta, además, al tiempo de tránsito en pista: una vez que un avión *pequeño* aterriza, ningún otro puede usar la misma pista durante 10 minutos; desde que aterriza un avión *mediano* hasta que cualquier otro puede usar la misma pista deben transcurrir 25 minutos. Por último, un avión grande usa la pista elegida para el aterrizaje durante 45 minutos.

Además el aeropuerto cobra tasas diferentes a los aviones según su tamaño: los aviones *grandes* pagan hasta 2.500 €; los *medianos* pagan 1.500 €, mientras que los *pequeños* pagan sólo 1.000 €.

(a) **(1 punto)** La dirección del aeropuerto está interesada en calcular el ingreso máximo que podría percibir por el uso de todas las pistas disponibles *por día* para un número arbitrariamente grande de aviones de cada tamaño. Debe asumirse que el aeropuerto está abierto las 24 horas del día.

Se pide modelar el problema como un problema de Programación Lineal. *Es imprenscindible expresar el modelo final en forma estándar de maximizacion* indicando claramente las *variables de decisión* elegidas y su significado.

(b) **(1 punto)** Además, la dirección del aeropuerto desea usar técnicas de Programación Lineal para decidir el orden de aterrizaje de los aviones que solicitan hacerlo a la torre de control con el objetivo de minimizar el tiempo total de espera de los aviones en el aire.

La torre de control asigna pistas de aterrizaje después de realizar algunas verificaciones de modo que lo hace cada 3 minutos. Mientras tanto, los aviones deben esperar en el espacio aéreo del aeropuerto.

Considérese el caso de tres aviones (uno *grande* y dos *medianos*) que han solicitado permiso de aterrizaje simultaneamente y que la primera pista de aterrizaje no está disponible porque está sometida a tareas de mantenimiento.

Modelizar el problema de decisión que consiste en determinar el orden de aterrizaje en las pistas disponibles de modo que el tiempo total de espera en el aire para estos tres aviones sea el mínimo. *Es imprenscindible expresar el modelo final en forma estándar de maximización* indicando claramente las *variables de decisión* elegidas y su significado.

Pregunta 2 (2 puntos)

Considérese la fórmula en Formal Normal Conjuntiva $F = \bigwedge_{i=1}^{6} C_i$ que contiene las siguientes cláusulas:

$$\begin{array}{ll} C_1\colon (\overline{x}_1 \vee x_4 \vee \overline{x}_5) & C_4\colon (\overline{x}_2 \vee x_3) \\ C_2\colon (x_2 \vee \overline{x}_4 \vee x_5) & C_5\colon (\overline{x}_3) \\ C_3\colon (x_1) & C_6\colon (\overline{x}_6 \vee \overline{x}_1) \end{array}$$

Se pide responder razonadamente las siguientes cuestiones:

(a) **(½ puntos)** Indica qué literales son puros y resuelve la *resolución* de la fórmula F respecto de esos literales

(b) **(1 punto)** Obtener un modelo que satisfaga la fórmula resultante del apartado anterior con el algoritmo de Davis-Putnam. *Considera, para ello, las variables en orden ascendente de su subíndice.*

(c) **(½ puntos)** A partir de los resultados del algoritmo de Davis-Putnam, ¿cuántos modelos hay que satisfacen la fórmula del apartado anterior?

Pregunta 3 (3 puntos)

Considerése el siguiente problema de Programación Lineal:

$$\begin{array}{c} \text{mín } z = -2x_1 - 3x_2 - 4x_3 \\ \begin{array}{rcrcrcr} 3x_1 & + & 2x_2 & + & x_3 & \leq & 10 \\ 2x_1 & + & 5x_2 & + & 3x_3 & \leq & 15 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ puntos)** Expresar el problema de Programación Lineal anterior en forma *estándar* de maximización.

(b) **(½ puntos)** Preparar el problema de Programación Lineal que ha resultado del apartado anterior para su aplicación con el algoritmo SIMPLEX para garantizar que pueda comenzarse con la matriz identidad.

(c) **(½ puntos)** Resolver el problema de Programación Lineal obtenido en el apartado anterior con el algoritmo SIMPLEX.

Es imprenscindible indicar claramente, en cada iteración: las variables escogidas en la base, su valor, y el valor de la función objetivo

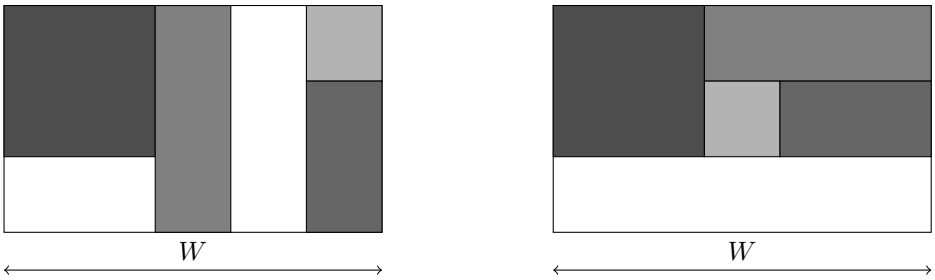
(d) **(½ puntos)** Interpretar el resultado y explicar qué conclusiones pueden extraerse de él.

(e) **(½ puntos)** ¿Cuál es el problema dual del problema de Programación Lineal original?

(f) **(½ puntos)** Calcula la contribución por unidad de recurso del problema de Programación Lineal original al valor óptimmo de la función objetivo.

Pregunta 4 (3 puntos)

Una empresa textil desea optimizar la cantidad de tela que debe emplearse para cortar hasta N patrones rectangulares diferentes de dimensiones $a_i \times b_i$, medidas en centímetros. Los cortes se realizan en un rollo de tela que tiene un ancho fijo de W centímetros y que, a todos los efectos, puede considerarse infinitamente largo. El corte de los patrones es siempre paralelo a los bordes del rollo y, si se desea, un patrón puede cortarse inmediatamente al lado de otro —esto es, sin dejar ningún hueco.



Las figuras anteriores muestran dos formas diferentes de cortar cuatro patrones de diferentes dimensiones indicados en diferentes tonos de gris. La figura de la izquierda es claramente ineficiente porque al cortar los patrones se utiliza un área mayor del rollo inicial de la que hacía falta como se muestra en la figura de la derecha.

Se ha decidido resolver el problema empleando *algoritmos de búsqueda*. Se pide:

(a) **(½ puntos)** Representar el problema como un *espacio de estados*

(b) **(½ puntos)** Asumiendo que los cortes se hacen con una precisión de centímetros, ¿Cuál es el factor de ramificación y profundidad máxima desarrollada por un algoritmo de búsqueda no informado?

(c) **(½ puntos)** Sugerir razonadamente el algoritmo de búsqueda no informada que mejor se adapta a este problema.

(d) **(1 punto)** Diseñar una función heurística $h(n)$ que sea *admisible* y que esté bien informada.

(e) **(½ puntos)** Sugerir razonadamente el algoritmo de búsqueda heurística que mejor se adapta a este problema.

Soluciones del examen de Heurística y Optimización Enero 2014

Problema 1

1. Como de costumbre, para la modelización de problemas de Programación Lineal, los pasos consisten en: uno, identificación de las *variables de decisión*; dos, formulación de las restricciones; tres, formulación de la función objetivo. Además será preciso hacer al final consideraciones adicionales sobre el tipo de problema a resolver —por ejemplo, si es de Programación Entera o no, y si es así, de qué tipo.

   Variables de decisión Puesto que el problema advertía explícitamente que el cálculo se hiciera para un número arbitrariamente grande de aviones, las variables a usar serán precisamente el número de aviones de cada tamaño que el aeropuerto debería atender (con las restricciones del problema) para maximizar sus ingresos en tasas. Ahora bien, como los aviones pueden aterrizar en diferentes pistas, son variables de decisión del problema las siguientes:

   x_i^G : número de aviones *grandes* que aterrizan en la pista número i
   x_i^M: número de aviones *medianos* que aterrizan en la pista número i
   x_i^P : número de aviones *pequeños* que aterrizan en la pista número i

   Como los aviones grandes sólo pueden aterrizar en la tercera pista, el primer grupo de variables contiene únicamente a x_3^G. Sin embargo, como los aviones medianos pueden aterrizar en cualquiera, el segundo grupo de variables está formado por: x_1^M, x_2^M y x_3^M. Por último, los aviones pequeños pueden hacerlo en una cualquiera de las dos primeras pistas y, por lo tanto, el último grupo de variables se descompone en las siguientes: x_1^P y x_2^P.

   En total, hay 6 variables de decisión. Como el número de aviones debe ser necesariamente un valor entero no negativo (es decir, $\mathbf{x} \in \mathbb{N}$) y todas las variables se refieren al número de aviones de diferentes tipos, se trata de un problema de Programación Entera Pura.

   Restricciones Las restricciones del problema son de dos tipos: de una parte relativas al uso de las pistas de aterrizaje disponibles y, de la otra, relativas al tiempo máximo de utilización de las pistas. Como todas ellas se refieren a las pistas de aterrizaje, sólo es necesario crear una restricción por pista como se indica a continuación. Nótese que como la dirección del aeropuerto está interesada en calcular el beneficio máximo en un día, el uso de las pistas no debería exceder las 24 horas que hay en un día. Ahora bien, como el tiempo de tránsito se da en minutos, el recurso de ocupación de las pistas será de $60 \times 24 = 1440$ minutos:

   La pistas número 1 y 2 pueden acoger tanto aviones pequeños como medianos de modo que se describen de la misma manera y se distinguen únicamente por las variables escogidas en cada caso (o relativas a la primera pista, o a la segunda):

   $$\begin{array}{rcrcl} 10x_1^P & + & 25x_1^M & \leq & 1440 \\ 10x_2^P & + & 25x_2^M & \leq & 1440 \\ & & \mathbf{x} \in \mathbb{N} & & \end{array}$$

   Análogamente, la tercera pista puede acoger aviones medianos o grandes, de modo que resulta:

   $$\begin{array}{rcrcl} 25x_3^M & + & 45x_3^G & \leq & 1440 \\ & & \mathbf{x} \in \mathbb{N} & & \end{array}$$

   Nótese como, en todos los casos, el término izquierdo de cada restricción sirve para calcular el tiempo de ocupación de la pista con cualesquiera tipos de aviones que pueden usarla y el término derecho establece un límite superior. Es especialmente importante observar que, como ya se ha anticipado, las restricciones se crean *por pista* y que estas no puedan agregarse. En otras palabras, la restricción:

   $$\begin{array}{rcrcl} 10x_1^P & + & 25x_1^M & \leq & 2880 \\ & & \mathbf{x} \in \mathbb{N} & & \end{array}$$

es un gravísimo error porque SIMPLEX podría usar entonces una pista durante dos días consecutivos y ni un sólo minuto la otra. Tampoco se solucionaría el problema con variables de decisión genéricas como x^P y x^M porque estas no servirían en absoluto para modelizar la ocupación de las pistas.

Función objetivo Se trata de un problema de maximización de beneficios, de modo que tendremos una función del tipo máx z. En particular, el término que se desea maximizar es el ingreso por tasas con 2.500 € por avión grande, 1.500 € por avión mediano y 1.000 € por avión pequeo. Por lo tanto, a partir de la definición de variables de decisión anterior resulta la función objetivo:

$$\text{máx } z = 2500x_3^G + 1500\left(\sum_{i=1}^{3} x_i^M\right) + 1000\left(\sum_{i=}^{2} x_i^P\right)$$

Finalmente, el problema completo de Programación Lineal diseñado de esta manera, queda como sigue:

$$\begin{array}{l} \text{máx } z = 2500x_3^G + 1500\left(\sum\limits_{i=1}^{3} x_i^M\right) + 1000\left(\sum\limits_{i=1}^{2} x_i^P\right) \\ \begin{array}{rcrcl} 10x_1^P & + & 25x_1^M & \leq & 1440 \\ 10x_2^P & + & 25x_2^M & \leq & 1440 \\ 25x_3^M & + & 45x_3^G & \leq & 1440 \end{array} \\ \mathbf{x} \in \mathbb{N} \end{array}$$

que aún no está en forma estándar de maximización como se pedía en el enunciado. Para ello, basta simplemente con añadir *variables de holgura* para convertir cada desigualdad en una igualdad —la solución del Problema 3 discute cómo se usan las variables de holgura además de ofrecer una discusión sobre la forma estándar de maximización:

$$\begin{array}{l} \text{máx } z = 2500x_3^G + 1500\left(\sum\limits_{i=1}^{3} x_i^M\right) + 1000\left(\sum\limits_{i=}^{2} x_i^P\right) \\ \begin{array}{rcrcrcrcl} 10x_1^P & + & 25x_1^M & + & s_1 & & & = & 1440 \\ 10x_2^P & + & 25x_2^M & & & + & s_2 & = & 1440 \\ 25x_3^M & + & 45x_3^G & & & & & = & 1440 \end{array} \\ \mathbf{x} \in \mathbb{N} \end{array}$$

donde s_1, s_2 y s_3 son las variables de holgura introducidas.

2. Se procederá en este caso como en el del apartado anterior, determinando una por una, las diferentes componentes que conforman la definición de un problema de Programación Lineal.

Variables de decisión En este caso, se trata de determinar el orden de llegada de los aviones a las pistas. Por lo tanto, en vez de tener variables de decisión para calcular el número de los aviones que llegan a cada pista, se registra el orden de llegada de cada avión a cada pista como sigue:

x_{ij}^G : El avión *grande* (G) aterriza el i-ésimo en la pista j
$x_{ij}^{m_1}$: El primer avión *mediano* (m_1) aterriza el i-ésimo en la pista j
$x_{ij}^{m_2}$: El segundo avión *mediano* (m_2) aterriza el i-ésimo en la pista j

Aunque hay tres aviones, $0 \leq j \leq 2$, resulta trivial observar que para reducir el tiempo de espera en el aire se deben usar ambas pistas simultaneamente. Por lo tanto, una vez que hayan aterrizado dos aviones al mismo tiempo, un tercero lo deberá hacer en alguna de las dos pistas —de hecho, en la primera que esté disponible. En otras palabras, $0 \leq j \leq 1$

Por lo tanto, hay hasta 10 variables de decisión en este problema que resultan de instanciar el subíndice i por 0 ó 1, para cada avión, en las pistas, j, que puede usar: x_{03}^G y x_{13}^G para el avión *grande*; x_{02}^{m1}, x_{12}^{m1}, x_{03}^{m1} y x_{13}^{m1} para el avión *mediano* m_1. Análogamente, x_{02}^{m2}, x_{12}^{m2}, x_{03}^{m2} y x_{13}^{m2} para el segundo avión mediano.

Una decisión importante es el dominio de cada una de estas variables. Puesto que sólo determinan el orden de aterrizaje, deben modelizarse como valores 0–1 que se interpretan como que aterrizan en la pista j en la posición i positiva o negativamente, respectivamente.

Por lo tanto, se trata de un problema de programación entera pura 0–1.

Restricciones Las restricciones de este problema son triviales. Se trata, simplemente, de exigir que cada avión debe aterrizar. En otras palabras, cada una de las variables x_{ij}^G, xm_{1ij} y xm_{2ij} debe tomar el valor 1, pero sólo una de ellas. Esto es, todos los aviones deben aterrizar, pero sólo lo deben hacer una vez:

$$\begin{array}{l} \sum_i \sum_j x_{ij}^G = 1 \\ \sum_i \sum_j x_{ij}^{m_1} = 1 \\ \sum_i \sum_j x_{ij}^{m_2} = 1 \end{array}$$

donde se ha obviado la restricción

$$\mathbf{x} \in \{0, 1\}$$

Función objetivo La administración del aeropuerto requiere explícitamente minimizar el tiempo de espera en el aire de todos los aviones. Por lo tanto, la función objetivo resulta de calcular el tiempo de espera en el aire de cada avión para cada una de los diferentes eventos que pueden darse:

- Si el avión *grande* aterriza primero, entonces alguno de los aviones medianos deberá esperar 45 minutos si desea utilizar la pista 3.
- Si un avión *mediano* emplea una pista, entonces el tiempo de espera del que le sigue serán 25 minutos.

Puesto que el tiempo de espera afecta únicamente al segundo avión que desea utilizar una pista, la función objetivo consiste únicamente en sumar el tiempo de ocupación de las pistas de los que aterrizan en primer lugar:

$$\text{mín}\, z = 45 x_{13}^G + 25 \left(\sum_{j=1}^{2} x_{1j}^{m_1} + x_{1j}^{m_2} \right)$$

Y a estas alturas ya es posible, por fin, presentar la formulación completa del problema de Programación Lineal:

$$\begin{array}{rc} \text{mín}\, z = 45 x_{13}^G + 25 \left(\sum\limits_{j=1}^{2} x_{1j}^{m_1} + x_{1j}^{m_2} \right) & \\ \sum_i \sum_j x_{ij}^G = & 1 \\ \sum_i \sum_j x_{ij}^{m_1} = & 1 \\ \sum_i \sum_j x_{ij}^{m_2} = & 1 \\ \mathbf{x} \in \{0, 1\} & \end{array}$$

El enunciado, sin embargo, exigía que la solucón se expresara en forma estándar de *maximización*. Para ello, basta con cambiar el signo de la función objetivo:

$$\begin{array}{rc} \text{máx}\, z = -45 x_{13}^G - 25 \left(\sum\limits_{j=1}^{2} x_{1j}^{m_1} + x_{1j}^{m_2} \right) & \\ \sum_i \sum_j x_{ij}^G = & 1 \\ \sum_i \sum_j x_{ij}^{m_1} = & 1 \\ \sum_i \sum_j x_{ij}^{m_2} = & 1 \\ \mathbf{x} \in \{0, 1\} & \end{array}$$

Problema 2

1. Un literal se define como la asociación de una variable (por ejemplo, x_1) y su signo. Por lo tanto, x_1 y $\overline{x}_1$ son literales distintos aunque se refieren a la misma variable. Negando cualquiera de ellos se obtiene el otro.

   Un literal ℓ es puro si y sólo si no aparece negado en ninguna cláusula. Todas las variables de la fórmula conjuntiva F aparecen negadas y afirmadas menos x_6 que aparece negada únicamente.

   Por otra parte, la resolución de una fórmula F respecto de un literal puro ℓ es $F \setminus \ell$ que consiste en las mismas cláusulas originales de F menos aquellas que contiene el literal puro ℓ. El resultado de la resolución es razonable puesto que buscando modelos que satisfagan expresiones lógicas en Forma Normal Conjuntiva, las cláusulas que contengan un literal puro se satisfacen automáticamente en cuanto la variable correspondiente toma el valor que se corresponde con el signo del literal. En este caso, $x_6 = \bot$.

   La fórmula resultante, $F' = Res(F, x_6)$, se muestra en el siguiente apartado.

2. El enunciado advertía, intencionadamente, que se aplicara la resolución considerando las variables en orden ascendente de su subíndice. Esto es, primero x_1, luego x_2 y así sucesivamente. Por lo tanto, a continuación se aplica el algoritmo con este orden de variables a la fórmula conjuntiva F' que resulta de eliminar la cláusula que contenía el único literal puro que se encontró en el apartado anterior:

   $$\begin{array}{ll} C_1\colon (\overline{x}_1 \vee x_4 \vee \overline{x}_5) & C_4\colon (\overline{x}_2 \vee x_3) \\ C_2\colon (x_2 \vee \overline{x}_4 \vee x_5) & C_5\colon (\overline{x}_3) \\ C_3\colon (x_1) & \end{array}$$

   donde $F' = \bigwedge_{i=1}^{5} C_i$.

   A continuación se detallan todos los pasos del algoritmo de Davis-Putnam. Primero se aplica el procedimiento de resolución *hacia delante*. Si y sólo si resultara el conjunto vacío, entonces se aplicará la búsqueda del modelo hacia atrás.

   Paso 0 $G_0 = C_i\Big|_{i=1}^{5}$

   Tal y como adverta el enunciado, primero se elige la variable x_1 y se almacena la selección en un vector dedicado `varSelect`. Además, el vector `layerSeq` contendrá las cláusulas involucradas en cada paso:

   $$\begin{array}{ll} \text{varSelect}[0] = x_1 & \textit{varSelect almacena la seleccion} \\ & \textit{de variables por paso, y} \\ \text{layerSeq}[0] = G_0 \setminus \text{Res}(G_0, x_1) = & \textit{layerSeq las clausulas} \\ \quad G_0 \setminus \{C_2, C_4, C_5, C_6 :\ (x_4 \vee \overline{x}_5)\} = & \textit{la resolucion crea } C_6 \\ \quad \{C_1, C_3\} & \end{array}$$

   La ventaja de calcular las cláusulas involucradas en cada paso como una diferencia de conjuntos es que, al mismo tiempo se obtiene el conjunto de cláusulas que deben considerarse en el siguiente paso:

   Paso 1 $G_1 = \{C_2, C_4, C_5, C_6\}$

   A continuación se considera la siguiente variable en el orden indicado, x_2

   $$\begin{array}{l} \text{varSelect}[1] = x_2 \\ \text{layerSeq}[1] = G_1 \setminus \text{Res}(G_1, x_2) = \\ \quad G_1 \setminus \{C_5, C_6, C_7 :\ (x_3 \vee \overline{x}_4 \vee x_5)\} = \\ \quad \{C_2, C_4\} \end{array}$$

Paso 2 $G_2 = C_i\Big|_{i=5}^{7}$

La siguiente variable a considerar es x_3

$\text{varSelect}[2] = x_3$
$\text{layerSeq}[2] = G_2 \setminus \text{Res}(G_2, x_3) =$
$G_2 \setminus \{C_6, C_8 :\ (\overline{x}_4 \vee x_5)\} =$
$\{C_5, C_7\}$

Paso 3 $G_3 = \{C_6, C_8\}$

En el último paso, se considera la variable x_4 como sigue:

$\text{varSelect}[2] = x_4$
$\text{layerSeq}[2] = G_3 \setminus \text{Res}(G_3, x_4) =$
$G_3 \setminus \varnothing =$ *Puesto que resulta una tautologia*
$\{C_6, C_8\}$

El motivo por el que en el último paso no se ha generado nada es porque la resolución de las cláusulas $C_6 :\ (x_4 \vee \overline{x}_5)$ y $C_8 :\ (\overline{x}_4 \vee x_5)$ respecto de x_4 generaría la cláusula $(x_5 \vee \overline{x}_5)$ que es una tautología y, por ese motivo, no se añade.

Como el proceso de resolución hacia delante ha concluido con el conjunto vacío se sabe que la fórmula F' es satisfacible. Ahora se procede, en el segundo paso, a la búsqueda de un modelo que satisfaga la Forma Normal Conjuntiva F'.

En el primer paso (hacia atrás), deben satisfacerse las clásulas $\text{layerSeq}[3] = \{C_6, C_8\}$ dando un valor a la variable x_4. Por ejemplo, haciendo $x_4 = \top$ la primera cláusula se satisface inmediatamente, pero no la segunda, que sólo puede satisfacerse ahora haciendo $x_5 = \top$.

A continuación, deben satisfacerse las cláusulas $\text{layerSeq}[2] = \{C_5, C_7\}$ considerando las asignaciones anteriores ($x_4 = \top$ y $x_5 = \top$) y, si hiciera falta, asignando un valor a la variable x_3. Efectivamente C_5 sólo puede satisfacerse con $x_3 = \bot$. Por otra parte, como $x_5 = \top$, C_7 ya está satisfecha.

En el penúltimo paso hacia atrás, deben satisfacerse las clausulas en $\text{layerSeq}[1]$, que son C_2 y C_4. La primera ya está satisfecha con la asignación $x_5 = \top$. Por otra parte, C_4 sólo puede satisfacerse con $x_2 = \bot$ —puesto que en el paso anterior se había elegido el valor $\bot$ para x_3.

Finalmente, deben considerarse las cláusulas involucradas en el primer paso hacia delante, esto es, C_1 y C_3. C_1 ya está satisfecha con el valor $x_4 = \top$ y C_3 sólo se puede satisfacer de una manera, haciendo $x_1 = \top$.

La siguiente tabla muestra el resultado de todos los pasos seguidos en la segunda fase del algoritmo de Davis-Putnam:

Iteración	3	2	1	0
varSelect^{-1}	x_4	x_3	x_2	x_1
layerSeq^{-1}	$\{C_6, C_8\}$	$\{C_5, C_7\}$	$\{C_2, C_4\}$	$\{C_1, C_3\}$
Asignación	$x_4 = \top, x_5 = \top$	$x_3 = \bot$	$x_2 = \bot$	$x_1 = \top$

De modo que la fórmula F' se satisface con el modelo:

$$M = \{x_1 = \top, x_2 = \bot, x_3 = \bot, x_4 = \top, x_5 = \top\}$$

Aunque no se pedía explícitamente en el enunciado, es muy fácil extender este modelo para obtener un modelo de la fórmula original F. Como $\overline{x}_6$ era un literal puro, basta con añadir $x_6 = \bot$, resultando:

$$M = \{x_1 = \top, x_2 = \bot, x_3 = \bot, x_4 = \top, x_5 = \top, x_6 = \bot\}$$

3. Repitiendo el mismo procedimiento del segundo paso en el apartado anterior resulta fácil advertir que la asignación $x_4 = \top$ era arbitraria. No sólo eso, de esa decisión se desprendieron las asignaciones para el resto de variables observando que eran asignaciones *forzosas*. Por lo tanto, empezando con $x_4 = \top$ hay un modelo que satisface la fórmula proposicional del primer apartado.

   ¿Qué sucede haciendo inicialmente $x_4 = \bot$? Como esta asignación se hace en el primer paso de la segunda parte del algoritmo de Davis-Putnam, las cláusulas que deben satisfacerse son C_6 y C_8. Con $x_4 = \bot$, se satisface C_8 y ahora necesariamente $x_5 = \bot$ para poder satisfacer C_6. A partir de este punto, las observaciones que siguen para concluir el modelo son exactamente las mismas que antes, de modo que resultaría el modelo:

$$M = \{x_1 = \top, x_2 = \bot, x_3 = \bot, x_4 = \bot, x_5 = \bot\}$$

   En otras palabras, comenzando con $x_4 = \bot$ se obtiene un único modelo. Como quiera que empezando con $x_4 = \top$ sólo se obtenía uno también y x_4 sólo puede tomar los valores $\bot$ o $\top$, hay entonces dos modelos en total que satisfacen la fórmula proposicional del primer apartado.

Problema 3

Este problema es, exactamente, el mismo ejercicio que se presenta en la Wikipedia bajo el epígrafe ALGORITMO SIMPLEX tanto en las versiones en inglés (`http://en.wikipedia.org/wiki/Simplex_algorithm`), como en español (`http://es.wikipedia.org/wiki/Algoritmo_simplex`)

1. En el primer apartado se pedía expresar el problema de Programación Lineal del enunciado en *forma estándar* (de maximización), precisamente en preparación a su resolución con el algoritmo SIMPLEX.

   Un problema de programación lineal está en forma *estándar* si todas las restricciones son de igualdad, las variables de decisión son no negativas y, por último, el vector de constantes o recursos $\mathbf{b}$ no contiene términos negativos. Estará, adems, en forma de maximización si la función objetivo maximiza y de minimización en otro caso. El problema, tal y como estaba enunciado, ya verifica todas estas condiciones salvo la primera y última puesto que ninguna restriccin es del tipo de igualdad y la función objetivo es de minimización. Conviene aquí recordar:

   - Una restricción de la forma $\leq$ está acotada superiormente. Puesto que ninguna variable de decisión puede tomar valores negativos, es preciso *sumar* una *variable de holgura* para forzar la igualdad.
   - Análogamente, las restricciones de la forma $\geq$ estn acotadas inferiormente de modo que, con variables de decisión que no pueden tomar valores negativos, es preciso *restar* una *variable de holgura* para forzar la igualdad.

   Por lo tanto, el problema de Programación Lineal queda, como sigue, en forma estándar de maximización:

$$\begin{array}{c} \max z = 2x_1 + 3x_2 + 4x_3 \\ \begin{array}{rcrcrcrcrcl} 3x_1 & + & 2x_2 & + & x_3 & + & x_4 & & & = & 10 \\ 2x_1 & + & 5x_2 & + & 3x_3 & & & + & x_5 & = & 15 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

   donde, además, se ha cambiado el signo de la función objetivo multiplicando por -1.

2. No es preciso hacer ninguna transformación adicional puesto que los vectores columna $\mathbf{a_4}$ y $\mathbf{a_5}$ sirven para crear la matriz identidad de dimensión 2 como se requerirá, en el siguiente apartado, para poder aplicar el algoritmo SIMPLEX.

3. El algoritmo del SIMPLEX consiste en la aplicación iterativa de tres pasos: cálculo de las variables básicas, selección de la variable de entrada y selección de la variable de salida hasta que se detecte alguna de las siguientes condiciones:

- El problema puede mejorar el valor de la función objetivo indefinidamente. Se dice entonces que el problema está *no acotado*. Este caso se detecta cuando todas las componentes y_i de la variable de decisión x_i elegida para entrar en la base son todos negativos o nulos.
- El problema es irresoluble. Esto ocurre cuando en el segundo paso, todos los costes reducidos son positivos y el primer paso asignó un valor no negativo a alguna variable artificial.
  Este caso es, en este problema particular, imposible puesto que no hay variables artificiales. En otras palabras, la ausencia de variables artificiales garantiza la solubilidad del problema original.
- Se alcanza una solución factible y puede demostrarse que no es posible mejorarla. Esta condición se detecta como en el segundo caso pero cuando las variables artificiales (si las hubiera) tienen valores nulos.

Paso 0 Cálculo de una solución factible inicial

a) Cálculo de las variables básicas
La primera iteración se inicia con una base igual a la matriz identidad de dimensión 2, tal y como se pedía en el enunciado. Por lo tanto, son variables bsicas en este paso $\{x_4, x_5\}$

$$B_0 = I_2 \qquad B_0^{-1} = I_2$$
$$x_0^* = B_0^{-1}b = b = \begin{pmatrix} 10 \\ 15 \end{pmatrix} \quad z_0^* = c_{B_0}^T x_0^* = \begin{pmatrix} 0 & 0 \end{pmatrix}\begin{pmatrix} 10 \\ 15 \end{pmatrix} = 0$$

b) Selección de la variable de entrada
En las expresiones siguientes el clculo de los vectores y_i se ha embebido en el cálculo de los *costes reducidos* directamente:

$$z_1 - c_1 = c_{B_0}^T B_0^{-1} a_1 - c_1 = \begin{pmatrix} 0 & 0 \end{pmatrix} I_2 \begin{pmatrix} 3 \\ 2 \end{pmatrix} - 2 = -2$$
$$z_2 - c_2 = c_{B_0}^T B_0^{-1} a_2 - c_2 = \begin{pmatrix} 0 & 0 \end{pmatrix} I_2 \begin{pmatrix} 2 \\ 5 \end{pmatrix} - 3 = -3$$
$$z_3 - c_3 = c_{B_0}^T B_0^{-1} a_3 - c_3 = \begin{pmatrix} 0 & 0 \end{pmatrix} I_2 \begin{pmatrix} 1 \\ 3 \end{pmatrix} - 4 = -4$$

Por lo tanto, como la variable no básica con el valor más negativo es x_3, ésta será la variable que entre en la siguiente iteración

c) Selección de la variable de salida
La regla de salida establece que debe salir aquella variable con el menor cociente x_i/y_{ij} donde x_i es la variable elegida en el paso anterior para aadirse a la base y $0 \leq j < 2$ puesto que la base tiene dimensin 2:

$$\text{mín}\left\{\frac{10}{1}, \frac{15}{3}\right\}$$

y sale la variable x_5 que es la que se corresponde con la segunda fracción, precisamente la que tiene el menor valor.

Paso 1 Mejora de la solución actual (iteración #1)

a) Cálculo de las variables básicas
A continuación se mejora la calidad de la solución anterior. Las nuevas variables básicas son $\{x_3, x_4\}$

$$B_1 = \begin{pmatrix} 1 & 1 \\ 3 & 0 \end{pmatrix} \qquad B_1^{-1} = \begin{pmatrix} 0 & \frac{1}{3} \\ 1 & -\frac{1}{2} \end{pmatrix}$$
$$x_1^* = B_1^{-1}b = \begin{pmatrix} 5 \\ 5 \end{pmatrix} \quad z_1^* = c_{B_1}^T x_1^* = \begin{pmatrix} 4 & 0 \end{pmatrix}\begin{pmatrix} 5 \\ 5 \end{pmatrix} = 20$$

b) Selección de la variable de entrada

$$z_1 - c_1 = c_{B_1}^T B_1^{-1} a_1 - c_1 = \begin{pmatrix} 4 & 0 \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{3} \\ 1 & -\frac{1}{3} \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} - 2 = \frac{2}{3}$$

$$z_2 - c_2 = c_{B_1}^T B_1^{-1} a_2 - c_2 = \begin{pmatrix} 4 & 0 \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{3} \\ 1 & -\frac{1}{3} \end{pmatrix} \begin{pmatrix} 2 \\ 5 \end{pmatrix} - 3 = \frac{11}{3}$$

$$z_5 - c_5 = c_{B_1}^T B_1^{-1} a_5 - c_5 = \begin{pmatrix} 4 & 0 \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{3} \\ 1 & -\frac{1}{3} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} - 0 = \frac{4}{3}$$

Como todas las variables tienen costes reducidos estrictamente positivos, entonces es que ya hemos llegado a la solución óptima

En conclusión, el valor óptimo que resuelve el problema original es:

$$x^* = \begin{pmatrix} 0 \\ 0 \\ 5 \\ 5 \\ 0 \end{pmatrix} \quad z^* = 20$$

4. La interpretación de un problema incluye varias consideraciones como son estudiar: si el problema es o no satisfacible, si la solución es única o hay varias soluciones o si está o no acotado. Además, debe estudiarse el uso de recursos: si sobra o no alguno y cual es su contribución al crecimiento de la función objetivo.

   Interpretación de la solución De la solución se puede advertir lo siguiente:

   - El problema es factible porque la solución no contiene valores positivos para alguna variable artificial. En particular, de hecho ni siquiera hay variables artificiales en el problema.
   - La solución es única porque los costes reducidos son todos estrictamente positivos. Eso significa que cualquier cambio en la base implicaría un decremento neto en el valor de la función objetivo mientras que el caso de *soluciones múltiples* ocurre cuando una cantidad infinita de soluciones tiene el mismo valor de la función objetivo.
   - El valor de la función objetivo está acotado porque siempre se pudo aplicar la regla de salida con, al menos, alguna variable básica.

   Interpretación de los recursos De los recursos se advierte:

   - En la solución óptima se sugiere usar completamente el segundo recurso, mientras que del primero hay cinco unidades sobrantes, como lo atestiguan el valor *óptimo* de las variables de holgura x_5 y x_4 respectivamente.
   - Además, corresponde estudiar la contribución por cada unidad de recurso al crecimiento de la función objetivo. Esto, en particular, se hará en los dos siguientes apartados.

5. Para poder obtener el problema dual es preciso poner primero el problema de Programación Lineal en forma *simétrica* de maximización:

$$\begin{array}{c} \text{máx}\, z = 2x_1 + 3x_2 + 4x_3 \\ \begin{array}{rcrcrcl} 3x_1 & + & 2x_2 & + & x_3 & \leq & 10 \\ 2x_1 & + & 5x_2 & + & 3x_3 & \leq & 15 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

de donde resulta que el problema dual es:

$$\begin{array}{rcrcl} \text{mín}\, w = & 10x'_1 & + & 15x'_2 & \\ 3x'_1 & + & 2x'_2 & \geq & 2 \\ 2x'_1 & + & 5x'_2 & \geq & 3 \\ x'_1 & + & 3x'_2 & \geq & 4 \\ & & \mathbf{x} \geq \mathbf{0} & & \end{array}$$

6. Para la resolución del último apartado, basta con recordar la *interpretaciń económica* de las soluciones de un problema dual que advierte que:

> La variable dual x'^*_i indica la contribución por unidad del recurso i-ésimo b_i a la variación el en valor óptimo z^* actual del objetivo

Puesto que el enunciado requiere la contribución unitaria de todos los recursos, entonces es preciso calcular la solución completa del problema dual.

Para ello, es posible iniciar la aplicación de otro SIMPLEX. Sin embargo, en su lugar es preferible hacer uso del siguiente resultado teórico:

> Si el problema de programación lineal en forma simétrica tiene una solución óptima correspondiente a una base $\mathbf{B}$, entonces $\mathbf{x'}^T = \mathbf{c}_\mathbf{B}^T\mathbf{B}^{-1}$ es una solución óptima para el problema dual

En este teorema, los términos usados para el cálculo de la solución óptima del problema dual se refieren al problema primal, salvo que se indique explcitamente lo contrario. Por lo tanto c_B^T es el vector de costes de las variables básicas en la solución del problema primal y B la base usada para el cálculo de la misma solución —que se mostró en el último paso de aplicación del SIMPLEX. Por el contrario, x'^T es la solución del problema dual.

En particular:

$$x'^* = c_B^T B^{-1} = (4 \quad 0) \begin{pmatrix} 0 & \frac{1}{3} \\ 1 & -\frac{1}{3} \end{pmatrix} = \begin{pmatrix} 0 & \frac{4}{3} \end{pmatrix}$$

de donde resulta que la contribución del primer recurso al crecimiento de la función objetivo es nula, mientras que el del segundo recurso es de $\frac{4}{3}$ por unidad de recurso. No es una casualidad que el primero no contribuya en absoluto al crecimiento de la función objetivo puesto que, como ya se ha observado, hay hasta cinco unidades sobrantes del primer recurso, de modo que añadir más no hará crecer el valor de la función objetivo.

Problema 4

1. El espacio de estados es una formalización que habilita la aplicación de algoritmos de búsqueda (informados o no) para la resolución de problemas. Consiste únicamente, en la definición de estados y operadores que sirven para transitar entre ellos. Relacionando, después, el conjunto de posibles estados con otro de vértices V y el de operadores (convenientemente instanciados) con otro de arcos E, resulta entonces de forma natural la definición de un grafo, el *grafo de búsqueda* que se recorrerá eficientemente con el uso de *árboles de búsqueda*. Este ejercicio propone ejercitar todos estos conceptos y, en el primer apartado, el del espacio de estados.

   Estados Un estado en este problema está caracterizado por los patrones y el rollo de tela. Para su representación se podrían elegir estructuras (planas o, más eficientemente, clases) cuyos contenidos se discuten a continuación:

Rollo de tela El rollo de tela tiene un ancho fijo W que es un parámetro fundamental del problema (y que, por lo tanto, será usado más adelante) de modo que habrá un campo entero `W` que almacena el ancho del rollo.

Además, la cantidad de tela que se emplea es igual al área que se corta en total (esto es, sumando el área usada para extraer los patrones y también la que se desperdicia con los cortes). Como quiera que el área total será $W \times L$, entonces debe haber otro campo `L` que almacenará el largo que debe usarse en total en el rollo de tela. Este valor, a diferencia del anterior, debe actualizarse regularmente con cada corte en el rollo de tela.

Patrones Como de costumbre, cada patrón se identificará por un código único que les distingue de los demás. Para ello, se sugiere usar un campo alfanumérico `id` que lo almacena.

Además, en la especificación de la demanda, cada patrón debe ir acompañado de sus medidas que se almacenarán en atributos dedicados `ancho` y `largo`. Análogamente, la solución del problema debe almacenarse como la posición en la que se corta cada patrón. Para ello basta con usar dos atributos `x` e `y` que almacenen (por ejemplo) la esquina superior izquierda en la que se inicia el corte.

Por último, en la enumeración de soluciones parciales típicas de los algoritmos de búsqueda, es preciso mantener información actualizada de qué patrones han sido ya extraídos y cuáles no. Para ello, se usará un campo booleano `Cortado` que tendrá el valor `True` si lo ha sido y `False` en otro caso.

Operadores Este problema consiste en un único operador `cortar` que dado el código de un patrón y un par de coordenadas (x, y) practica el corte con las dimensiones del patrón especificado por su código en la posición indicada.

En la programación de este operador `cortar (id, x, y)` las *precondiciones* son: `Cortado=False`, $\texttt{x} \leq \texttt{W} - \texttt{ancho}$ y también el hecho de que el nuevo corte no interseca con ningún otro para el que `Cortado=True`. Los *efectos* son: `Cortado=True`, $\texttt{L} = \text{máx}\{\texttt{L}, \texttt{y} + \texttt{largo}\}$.

Una cuestión capital es el coste de cada corte. Como lo que quiere es minimizarse el área total, el coste sería el nuevo área requerida del rollo de tela del que debe disponerse cuando se hace el corte, esto es $c(n, n_i) = \texttt{L} - \text{máx}\{\texttt{L}, \texttt{y} + \texttt{largo}\}$ donde n es un nodo arbitrario y n_i es el sucesor suyo al que se transita con este operador. Obviamente, si $c(n, n_i)$ toma valores negativos, entonces es que no debe usarse ningún área adicional y entonces el coste es realmente 0. Por lo tanto, resulta mucho más apropiado decir:

$$c(n, n_i) = \text{máx}\{0, \texttt{L} - \text{máx}\{\texttt{L}, \texttt{y} + \texttt{largo}\}\}$$

Podria pensarse en la posibilidad de otro operador `extraer` cuya función sea la de tirar del rollo de tela. Pero esto es absolutamente innecesario. Puede pensarse, simplemente, que las acciones de corte se hacen sobre un rollo de tela arbitrariamente largo, precisamente como se advertía explícitamente en el enunciado. Una vez que el problema ha sido resuelto es entonces cuando se pone el área obtenida a disposición del sistema de corte.

2. Cualquiera de los algoritmos de búsqueda (informada o no) que pueden idearse para resolver este problema óptimamente deben decidir en cada nodo el punto (x, y) en el que realizan el corte. Además, cada operador `cortar` realizará un corte y se sabe que hay que hacer exactamente N.

   Por lo tanto:

   Factor de ramificación El máximo factor de ramificación se da cuando todo el ancho del rollo de tela está a disposición del corte del patrón con el mínimo `ancho`. En particular, como W es el ancho en centímetros y los cortes se hacen con una precisión también de centímetros, entonces la coordenada x de la esquina superior izquierda del corte del patrón más estrecho (que puede suponerse igual a $\texttt{ancho} = 1$) puede tomar uno de los valores en el dominio $\{0, W - 1\}$, de modo que es W.

   Alternativamente, este factor de ramificación máximo puede calcularse con más precisión a la vista de la información disponible de la demanda en cada estado (véase el apartado anterior)

y tomando, en vez de la unidad como el ancho más pequeño, el que efectivamente sea el más estrecho. Es decir, $b_{max} = W - \text{mín}_i\{\texttt{ancho}_i\}$.

Profundidad La profundidad máxima es, a propósito de las observaciones anteriores, estrictamente igual al número de cortes que deben realizarse. Como quiera que el enunciado advertía que había exactamente N patrones, entonces $d_{\text{máx}} = N$.

3. Como este es un problema con costes las primeras alternativas son *Ramificación y Acotación en Profundidad* y *Dijkstra*. Ahora bien, como la profundidad está acotada (de modo que después de recorrer un camino de longitud $d_{\text{máx}}$ se sabe con seguridad que se ha llegado a una solución), el algoritmo idóneo es, sin ningún lugar a dudas, *Ramificación y Acotación en Profundidad*.

4. La generación de heurísticas admisibles se sigue de la técnica de *relajación de restricciones*. En su aplicación, se observan las restricciones del problema y se relajan todas o un subconjunto de ellas hasta que es posible resolver el problema resultante de forma óptima. Como quiera que las restricciones del problema se encuentran típicamente en las *precondiciones* de los operadores del problema (estudiadas en el primer apartado) son relajaciones factibles las siguientes:

   a) La relajación más sencilla consiste en violar el ancho del rollo de tela y que los cortes no deben intersecar. Si se relajan estas dos restricciones, entonces los cortes podrían hacerse siempre en la misma esquina superior izquierda.

   En este caso, la solución óptima consistiría en un área que es igual a $W \times L_{max}$, donde L_{max} es el máximo largo de todos los patrones especificados:

   $$h_1(n) = W \times \max_i\{L_i\}$$

   esto es así, incluso si todos los patrones son mucho menos anchos que W porque el ancho, como se advertía, es fijo.

   b) Si sólo se relaja el ancho del rollo de tela de modo que pudiera cortarse más allá de los W centímetros, entonces el corte de los patrones se haría sin que intersecaran, uno junto al otro, disponiendo de un largo arbitrariamente infinito.

   Sin embargo, esta relajación daría la misma función heurística que antes:

   $$h_2(n) = h_1(n) = W \times \max_i\{L_i\}$$

   c) Es posible mejorar estas estimaciones (admisibles, es decir, que no sobreestiman la cantidad de área requerida) sin violar ninguna de esas dos restricciones simplemente asumiendo que toda la tela que hay en la cantidad usada en cualquier estado está toda *junta* a disposición de los siguientes cortes.

   En este caso,

   $$h_3(n) = \frac{1}{W}\left(A - A'\right)$$

   donde A es el área requerida por todos los cortes que aún están pendientes de hacerse y A' es el área no usada por los cortes que ya se han realizado. El primer término puede calcularse fácilmente como la suma de los productos del `ancho` y `largo` de cada patrón con `Cortado=False`. El segundo término se calcula, simplemente como la diferencia de $W - L - \sum \prod_i \texttt{ancho}_i \times \texttt{largo}_i$,

   para los patrones que ya han sido extraídos donde L es el máximo largo requerido por los cortes considerados hasta este momento.

5. La selección de un algoritmo de búsqueda informada para este caso depende, enteramente, de la dificultad del problema:

A* El algoritmo A* es admisible y garantiza, por lo tanto, que encontrará soluciones óptimas si la función heurística que lo guía también es admisible. Además, es un algoritmo rápido puesto que no reexpande nodos (y, con frecuencia, las ordenaciones de la lista abierta se pueden hacer en $O(1)$ con las estructuras de datos adecuadas si la función objetivo sólo toma valores enteros como es nuestro caso —véase el primer apartado). Sin embargo, tiene un consumo de memoria exponencial.

IDA* El algoritmo IDA* reexpande nodos pero no ordena nodos y, mucho más importante aún, tiene un consumo de memoria lineal en la profundidad de la solución (que en nuestro caso es siempre igual a N). Además, también es un algoritmo de búsqueda admisible.

Por lo tanto, para la resolución de instancias sencillas se podría sugerir el primer algoritmo pero, para las instancias más complicadas se recomienda el segundo y, en general, el segundo es el más apropiado si el problema que se pretende resolver no tiene muchas transposiciones.