Departamento de Informática
Universidad Carlos III de Madrid

Grado en Informática

Heurística y Optimización

22 de Enero de 2016

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 (1½ puntos)

Un grupo de n estudiantes comparten piso, y han acordado que cada noche hará la cena uno de ellos, y sólo uno. Por supuesto, cada uno de ellos tiene diferentes habilidades culinarias y, por ello, cada uno de sus compañeros gratificará económicamente con c_i euros al estudiante i que haga la cena. Además, cada estudiante tiene diferentes disponibilidades, de modo que cada uno cocinará b_i veces como máximo durante todo el curso.

Desean, por lo tanto, establecer un calendario para ver quién hará la cena cada noche durante los m días que dura el curso —donde, naturalmente, $m \geq n$.

Se pide responder razonadamente las siguientes preguntas:

(a) **(1 punto)** Modeliza este problema como un problema de programación lineal en el que se desea minimizar la cantidad total a pagar por todos los estudiantes.

(b) **(½ puntos)** Considera ahora que cada estudiante tiene una disponibilidad diferente por días. Por ejemplo, el estudiante 1 podría cocinar los lunes y miércoles, pero no el resto de días de la semana, y el resto de estudiantes podrían tener (o no) restricciones como ésa.

¿Qué cambios hay que hacer en el modelo del apartado anterior para considerar este nuevo tipo de restricciones? *Indica, únicamente, los cambios sobre la modelización del apartado anterior justificando tus decisiones.*

Pregunta 2 (1½ puntos)

Considera la fórmula F_1 en Forma Normal Conjuntiva formada por las siguientes cláusulas:

$$\begin{array}{ll} C_1:(x_1 \vee \bar{x}_2) & C_4:(x_1 \vee \bar{x}_4 \vee \bar{x}_5 \vee x_6) \\ C_2:(\bar{x}_1 \vee x_4 \vee x_7) & C_5:(x_3 \vee x_5) \\ C_3:(\bar{x}_2 \vee x_3 \vee \bar{x}_5) & C_6:(\bar{x}_3 \vee \bar{x}_6 \vee \bar{x}_7) \end{array}$$

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ punto)** ¿La fórmula F_1 tiene literales puros? Si fuera así, ¿qué fórmula F_1' resultaría de hacer la resolución respecto de ellos?

(b) (½ **puntos**) Aplicar Davis-Putnam (DP) para encontrar un modelo que satisfaga la fórmula F_1' obtenida en el paso anterior, en caso de que exista alguno.

Se exige aplicar DP usando las variables en orden ascendente de su subíndice

Considérese ahora, en su lugar, la fórmula F_2 en Forma Normal Conjuntiva formada por las siguientes cláusulas:

$$\begin{array}{ll} C_1{:}(x_1 \vee x_2 \vee x_3) & C_3{:}(\bar{x}_1 \vee \bar{x}_2 \vee x_3) \\ C_2{:}(\bar{x}_1 \vee x_2 \vee \bar{x}_3) & C_4{:}(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \end{array}$$

Se pide responder razonadamente las siguientes preguntas:

(c) (½ **puntos**) Aplicar Davis-Putnam-Logemann-Loveland (DPLL) para enumerar todos los modelos que satisfagan la fórmula F_2, en caso de que exista alguno.

Se exige aplicar DPLL usando las variables en orden ascendente de su subíndice

Pregunta 3 (1 puntos)

Considérese la red de restricciones (X, D, C) definida por:

$$\begin{array}{lcl} X & = & \{x_1, x_2, x_3\} \\ D_1 & = & \{1, 2, 3, 4\} \\ D_2 & = & \{a, b, c, d\} \\ D_3 & = & \{\text{rojo}, \text{verde}\} \\ R_{12} & = & \{(2, a), (3, b), (4, c)\} \\ R_{13} & = & \{(2, \text{rojo}), (3, \text{verde})\} \\ R_{23} & = & \{(b, \text{rojo}), (c, \text{verde})\} \end{array}$$

Se pide responder razonadamente las siguientes preguntas:

(a) (½ **puntos**) ¿Las variables x_1 y x_2 son *arco-consistentes*? Si o no, ¿es necesario hacer alguna modificación para que lo sean?

(b) (½ **puntos**) ¿Las variables x_1 y x_2 son *camino-consistentes* respecto de x_3? Si o no, ¿es necesario hacer alguna modificación para que lo sean?

Pregunta 4 (3 puntos)

Considerése el siguiente problema de Programación Lineal:

$$\begin{array}{rcrcrl} & & \multicolumn{4}{l}{\max z = 2x_1 - 3x_2} \\ - & 3x_1 & + & 2x_2 & \leq & 1 \\ & 2x_1 & + & x_2 & \leq & 4 \\ - & 2x_1 & + & 6x_2 & \geq & 3 \\ & & \multicolumn{4}{l}{\mathbf{x} \geq \mathbf{0}} \end{array}$$

Se pide responder razonadamente las siguientes preguntas:

(a) (½ **puntos**) Resolver el problema utilizando el método de *resolución gráfica*

(b) (½ **puntos**) Resolver el mismo problema utilizando el método de *resolución gráfica* pero considerando las restricciones adicionales de que x_1 y x_2 deben ser enteras.

(c) (½ **puntos**) Expresar el problema de Programación Lineal del primer apartado en forma *estándar* de maximización de modo que, además, sea posible iniciar la aplicación del algoritmo SIMPLEX con una base igual a la matriz identidad.

(d) **(1 punto)** Resolver el problema de Programación Lineal obtenido en el apartado anterior con el algoritmo SIMPLEX.

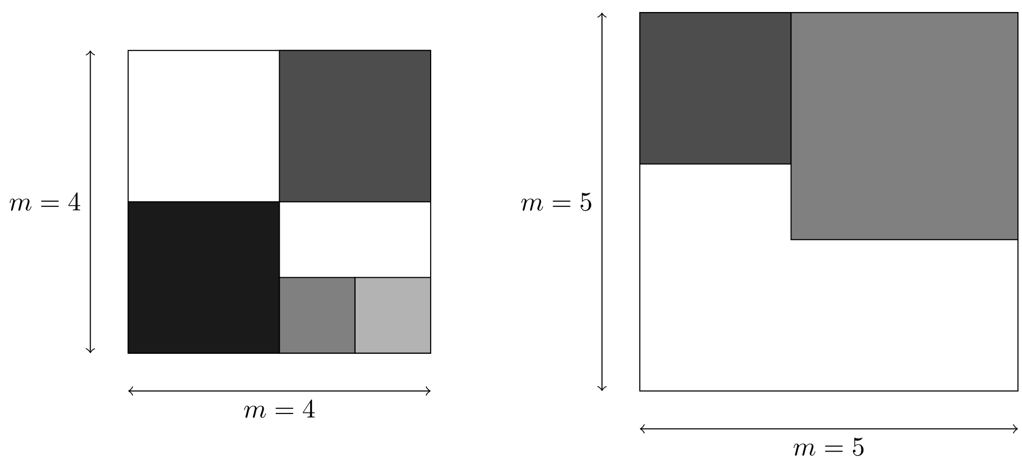
Es imprenscindible indicar claramente, en cada iteración: las variables escogidas en la base, su valor, y el valor de la función objetivo

(e) **($1/2$ puntos)** Interpretar las soluciones halladas y explicar qué conclusiones pueden extraerse.

Pregunta 5 (3 puntos)

Las zonas de carga de un buque carguero se dividen en regiones estrictamente cuadradas de dimensión $m \times m$ para satisfacer varias restricciones en relación con la distribución de pesos que debe soportar. Dada una cantidad arbitraria N de contenedores siempre cuadrados de dimensiones diferentes $l_i \times l_i$, $0 \leq i < N$, ¿cuál es la dimensión m del cuadrado *más pequeño* que inscribe los N contenedores?

A continuación se muestran dos ejemplos. En el caso de la izquierda, el cuadrado de menor área que inscribe dos cuadrados de 1×1 y otros dos de 2×2 es de 4×4. El segundo caso muestra que no es posible inscribir un cuadrado de 2×2 y otro de 3×3 en un cuadrado de menos de 5×5.



Se pide responder razonadamente las siguientes preguntas:

(a) **($1/2$ puntos)** Representar el problema como un *espacio de estados*

(b) **($1/2$ puntos)** ¿Qué profundidad tendría un árbol de búsqueda desarrollado por un algoritmo de búsqueda de fuerza bruta para resolver óptimamente este problema?

(c) **($1/2$ puntos)** Habida cuenta de que queremos encontrar soluciones óptimas, ¿qué algoritmo de búsqueda *no informada* sugerirías para su resolución?

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

(e) **($1/2$ puntos)** Habida cuenta que la función heurística $h(n)$ sea admisible, ¿qué algoritmo de búsqueda *heurística* es el más indicado para resolver este problema óptimamente?

Soluciones del examen de Heurística y Optimización Enero 2016

Problema 1

Este problema está adaptado de una pregunta realizada en `stackexchange.com`[1]. Básicamente, el cambio más importante ha consistido únicamente en añadir costes para que sea posible tener una función objetivo.

Se trata de un problema de *asignación* en el que deben asignarse estudiantes a días preservando las restricciones del problema. En lo sucesivo se asume que los días que dura el curso están numerados desde 1 hasta m y que los estudiantes se distinguen también por un número en el intervalo $[1, n]$.

1. En primer lugar se estudia cuáles serán las *variables de decisión*. A partir de ellas, se presentarán las restricciones y, por último, la función objetivo.

   Variables de decisión Se propone el uso de las variables de decisión:

   $x_{ij} \fallingdotseq 1$ si el estudiante i cocina el día j-ésimo y 0 en caso contrario

   de donde resulta obvio que se trata de un problema de programación entera 0–1.

   Restricciones Las restricciones del problema son, en el orden en que se enuncian, las siguientes:

   - "*Cada noche hará la cena uno de ellos, y sólo uno*". Por lo tanto, para todos los días debe ocurrir que la suma de las variables de decisión para ese mismo día y todos los estudiantes debe ser exactamente igual a 1:

   $$\sum_{i=1}^{n} x_{ij} = 1, \quad 1 \leq j \leq m$$

   - "*Cada uno cocinará b_i veces como máximo durante todo el curso*", de modo que para cada estudiante la suma de todos los valores de j para los que toma el valor 1 no puede exceder su disponibilidad:

   $$\sum_{j=1}^{m} x_{ij} \leq b_i, \quad 1 \leq i \leq n$$

   - Cómo se explicaba anteriormente, una restricción importantísima es que las variables de decisión únicamente pueden tomar los valores 0 ó 1:

   $$x_{ij} \in \{0, 1\}$$

   Función objetivo Puesto que el estudiante i cobra c_i euros a cada uno de sus $(n-1)$ compañeros cada vez que cocine, el desembolso total de los estudiantes será igual a $(n-1)c_i$. Como sólo cocinará un estudiante cada día, la expresión $\sum_{i=1}^{n}(n-1)c_i x_{ij}$ determina el desembolso total para el día j-ésimo independientemente de quien tenga que cocinar ese día. Por lo tanto, el desembolso total a lo largo del curso se calcula acumulando el gasto de todos los días:

   $$\min z = \sum_{j=1}^{m} \sum_{i=1}^{n} (n-1) c_i x_{ij}$$

   que, como se solicitaba en el enunciado, se debía minimizar.

   Obsérvese que el número de estudiantes, n es una constante del problema de modo que puede eliminarse sin riesgo de perder la solución óptima.

[1] Ver `http://cs.stackexchange.com/questions/51776/assign-m-tasks-to-n-workers-with-m-geq-n`

2. Para la segunda parte se sugiere usar un parámetro p_{ij} definido de la siguiente manera:

$$p_{ij} = \begin{cases} 0, & \text{si el estudiante } i \text{ puede cocinar el día } j\text{-ésimo} \\ 1, & \text{en otro caso} \end{cases}$$

   Nótese que la definición del parámetro podría parecer contraintuitiva porque vale 0 si el estudiante efectivamente puede cocinar ese día. El motivo es que la restricción que ahora se añadirá sirve para prevenir que se asigne un estudiante a un día en el que no puede cocinar:

$$\sum_{i=1}^{n} x_{ij} p_{ij} = 0, \quad 1 \leq j \leq m$$

   de modo que si un estudiante no puede cocinar un día en particular, el parámetro p_{ij} tomará el valor 1 y la única forma, entonces, de verificar la restricción consiste en hacer que la variable de decisión correspondiente x_{ij} sea cero.

   El resto de restricciones del primer apartado se preservan y la función objetivo no debe modificarse tampoco.

Problema 2

1. Un literal ℓ es puro si y sólo si su negado $\bar{\ell}$ no está presente en la fórmula de lógica proposicional. Se puede verificar, por inspección de la fórmula F_1 dada en el enunciado que todas las variables x_i aparecen afirmadas y negadas, salvo x_2 que aparece únicamente negada. Por lo tanto, el literal $\bar{x}_2$ es puro.

   El método de resolución establece que cuando un literal es puro pueden eliminarse todas las cláusulas de la fórmula proposicional que lo contienen, de modo que resulta la fórmula en lógica proposicional F_1' que consiste en la conjunción de todas las cláusulas, menos la primera y tercera:

$$\begin{array}{ll} C_2{:}(\bar{x}_1 \vee x_4 \vee x_7) & C_5{:}(x_3 \vee x_5) \\ C_4{:}(x_1 \vee \bar{x}_4 \vee \bar{x}_5 \vee x_6) & C_6{:}(\bar{x}_3 \vee \bar{x}_6 \vee \bar{x}_7) \end{array}$$

   Nótese que, en la fórmula resultante no hay más literales puros —lo que, sin embargo, podría haber ocurrido y, entonces se procedería de la misma manera con ellos.

2. Para aplicar Davis-Putnam, se escoge en cada iteración un literal y se aplica la resolución a la red de cláusulas actual respecto de ese literal. En cada paso, se guardan las variables usadas y las cláusulas involucradas de modo que si eventualmente resultara el conjunto vacío, $\varnothing$, se usa esa información para generar un modelo que valide la expresión inicial.

   A continuación se muestran todos los pasos del algoritmo DP. En cada paso se muestran la red de cláusulas G_i, la variable empleada x_j (que será, como se solicita en el enunciado, la siguiente en orden ascendiente de su subíndice), y las cláusulas involucradas, que se calculan como la diferencia entre la red de cláusulas de cada paso y el resultado de aplicar resolución: $G_i \backslash \text{Res}(G_i, x_j)$.

   Paso 0 $G_0 = \{C_2, C_4, C_5, C_6\}$

   La primera variable a utilizar es x_1 y el resultado de la resolución a las cláusulas de la red actual respecto de esta variable es:

$$\text{Res}(G_0, x_1) = \{C_5, C_6\}$$

   puesto que, x_1 no es un literal puro y aparece en las cláusulas C_2 y C_4, el método de resolución generaría una nueva cláusula con la disyunción de los literales que acompañan a x_1 en C_4 y a $\bar{x}_1$ en C_2: $(\bar{x}_4 \vee \bar{x}_5 \vee x_6 \vee x_4 \vee x_7)$ que, de hecho, es una tautología (porque x_4 aparece afirmado y negado) y, por lo tanto, no se genera. Por otra parte, las cláusulas C_5 y C_6 no contienen x_1 y, por ello, pasan inalteradas.

Por lo tanto, las cláusulas involucradas en este paso se calculan con la expresión:

$$G_0 \backslash \text{Res}(G_0, x_1) = G_0 \backslash \{C_5, C_6\} = \{C_2, C_4\}$$

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

En este paso, la red de cláusulas consiste en las cláusulas que resultaron de hacer la resolución en el paso anterior, esto es, $\{C_5, C_6\}$. La siguiente variable a usar será x_3 (puesto que x_2 fue eliminado en el apartado anterior):

$$\text{Res}(G_1, x_3) = \{C_7 : (x_5 \vee \bar{x}_6 \vee \bar{x}_7)\}$$

Como antes, x_3 no es un literal puro y aparece afirmado en C_5 y negado en C_6, de modo que el método de resolución genera una cláusula nueva que consiste en la disyunción de los literales que acompañan a un literal y otro en sus respectivas cláusulas. A diferecia del caso anterior, la cláusula resultante no es una tautología y, por lo tanto, se genera como una cláusula nueva, C_7.

En este paso, las cláusulas involucradas son:

$$G_1 \backslash \text{Res}(G_1, x_3) = G_1 \backslash \{C_7\} = \{C_5, C_6\}$$

Paso 2 $G_2 = \{C_7\}$

La cláusula C_7 sólo contiene las variables x_5, x_6 y x_7. Por lo tanto, la siguiente variable a usar será la primera, x_5 que, de hecho, es un literal puro y, por lo tanto:

$$\text{Res}(G_2, x_5) = \varnothing$$

puesto que al eliminar la cláusula que contiene a x_5, no quedan más cláusulas en la red actual, G_2. Como de costumbre, las cláusulas involucradas se calculan en este paso como sigue:

$$G_2 \backslash \text{Res}(G_2, x_5) = G_2 \backslash \varnothing = G_2 = \{C_7\}$$

Como el procedimiento anterior concluyó con el conjunto vacío $\varnothing$, entonces puede asegurarse que la fórmula proposicional original F_1' es satisfacible. Para generar un modelo, se consideran ahora las variables usadas y las cláusulas involucradas en orden inverso:

Paso	Variables	Cláusulas
2	x_5	$\{C_7\}$
1	x_3	$\{C_5, C_6\}$
0	x_1	$\{C_2, C_4\}$

y, en cada paso, se elige un valor para la variable usada de modo que se satisfagan las cláusulas involucradas en el mismo paso. Además, si fuera preciso, se deben asignar valores a otras variables, siempre con el propósito de satisfacer las cláusulas de cada paso.

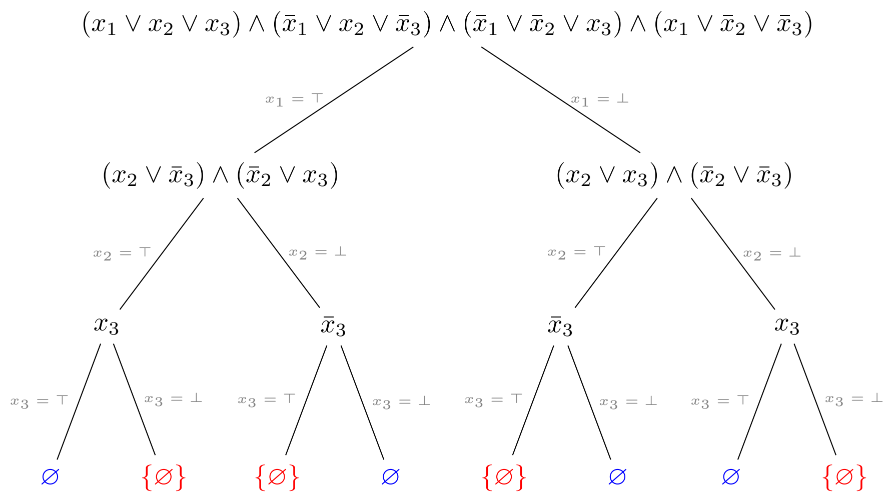
Inicialmente, si $x_5 = \top$, entonces C_7 se satisface. A continuación, como $x_5 = \top$, entonces C_5 ya está satisfecha. Para satisfacer C_6, basta con hacer $x_3 = \bot$. En el paso 0, ninguna de las asignaciones anteriores sirve para validar las cláusulas C_2 y C_4. Por lo tanto, para satisfacer la primera se hace $x_1 = \bot$, de modo que para satisfacer la segunda, y habida cuenta de que $x_5 = \top$, se puede hacer $x_4 = \bot$ (o, alternativamente, $x_6 = \top$). De esta forma, un modelo que satisface la fórmula proposicional F_1' es:

$$M_1 = \{x_1 = \bot, x_3 = \bot, x_4 = \bot, x_5 = \top\}$$

Nótese que otras variables no tienen valor asignado y, de hecho, pueden tomar cualquier valor sin afectar la satisfabilidad de la fórmula bajo el modelo presentado, M_1.

Por último, el modelo presentado es uno de tantos y otros modelos podrían haberse generado con el mismo procedimiento.

3. El algoritmo DPLL aplica *resolución unitaria* sobre la fórmula proposicional usando un literal que exista en ella, tanto afirmado, como negado. De esta forma, se construye un árbol (que, de hecho, se recorre *en profundidad* para evitar consumos exponenciales de memoria) en el que se enumeran progresivamente todas las asignaciones de verdadero ($\top$) o falso ($\perp$) a las variables de la fórmula proposicional original —mostradas en gris en la siguiente figura. Tal y como se advertía en el enunciado, las variables usadas se siguen en orden ascendente por su subíndice como se muestra a continuación:



Cada vez que la resolución unitaria genera el conjunto vacío (mostrado en azul en la figura), la fórmula inicial mostrada en la raíz del árbol generado por DPLL es satisfacible y el modelo resulta de las asignaciones que hay en el camino desde la raíz hasta el nodo terminal. Por el contrario, la cláusula vacía (mostrada en rojo en la figura) indica que el modelo intentado hasta ese nodo no satisface la fórmula original.

Por lo tanto, hay hasta cuatro modelos diferentes que satisfacen la fórmula proposicional F_2:

$$\begin{aligned} M_1 &= \{x_1 = \top, x_2 = \top, x_3 = \top\} \\ M_2 &= \{x_1 = \top, x_2 = \perp, x_3 = \perp\} \\ M_3 &= \{x_1 = \perp, x_2 = \top, x_3 = \perp\} \\ M_4 &= \{x_1 = \perp, x_2 = \perp, x_3 = \top\} \end{aligned}$$

Problema 3

1. La arco consistencia entre dos variables x_i y x_j sirve para determinar si para cada valor $a_i \in D_i$ existe otro $a_j \in D_j$ que satisfaga la restricción que relaciona las variables i y j, R_{ij}. Si no fuera así, entonces a_i puede eliminarse del dominio de la primera variable, D_i.

Por lo tanto, para verificar la arco-consistencia en toda la red de restricciones primero se observa cada par de variables y se eliminan los valores de cada dominio que sean arco inconsistentes. Al final de cada ronda, si ha habido cambios en los dominios se vuelve a empezar de nuevo hasta que por fin los dominios han alcanzado un *punto fijo*.

En este apartado se solicitaba verificar la arco-consistencia entre x_1 y x_2. Puesto que la arco-consistencia es *direccional*, se verifica a continuación en los dos sentidos:

$x_1 \to x_2$ Como se puede ver en el enunciado, existen valores de D_2 en R_{12} para los siguientes valores de D_1: 2, 3 y 4.

Por lo tanto, como no hay ningún valor en D_2 para $x_1 = 1$, este valor puede eliminarse del dominio de la primera variable, quedando ahora como sigue:

$$D_1 = \{2, 3, 4\}$$

$x_2 \to x_1$ De la misma manera, es fácil observar que existen valores de D_1 en R_{12} para los siguientes valores de D_2: a, b y c.

De modo que, puesto que no hay valores en D_1 para $x_2 = d$, este valor se elimina del dominio de la segunda variable resultando:

$$D_2 = \{a, b, c\}$$

Puesto que ha habido cambios en los dominios, ahora sería preciso volver a verificar la arco-consistencia entre las mismas variables. Sin embargo, es fácil verificar que no habría ningún cambio en los dominios.

Por lo tanto, la respuesta es que *sí* son arco-consistentes y la modificación que ha resultado de esta verificación ha sido eliminar el valor 1 de D_1 y d de D_2.

2. La camino consistencia (de longitud 2) entre dos variables x_i y x_j respecto de una tercera x_k, consiste en determinar si para cada asignación de valores $a_i \in D_i$ y $a_j \in D_j$ a las variables x_i y x_j respectivamente compatible con la restricción que las une, R_{ij}, existe un valor $a_k \in D_k$ que sea consistente con las relaciones R_{ik} y R_{jk}. Si no fuera así, la asignación (a_i, a_j) puede eliminarse de la relación R_{ij}.

   Por supuesto, la camino consistencia se aplica habida cuenta de que las variables x_i y x_j sean arco-consistentes. De hecho, ya se ha verificado la arco-consistencia entre las variables x_1 y x_2 en el apartado anterior, de modo que la verificación de la camino consistencia respecto de x_3 se hace ahora considerando todas las tuplas en R_{12} y si hay valores de x_3 que soportan cada restricción:

   $(2, a)$ Si $x_1 = 2$, entonces $x_3 = \text{rojo}$ puesto que $(2, \text{rojo})$ es la única tupla que aparece en R_{13} con $x_1 = 2$. Sin embargo, la tupla (a, rojo) no existe en R_{23}.

   Por lo tanto $(2, a)$ puede eliminarse de R_{12} puesto que es camino inconsistente con x_3.

   $(3, b)$ Como antes, si $x_1 = 3$, entonces necesariamente $x_3 = \text{verde}$ puesto que $(3, \text{verde})$ es la única tupla en R_{13} con $x_1 = 3$. Sin embargo, (b, verde) no existe en R_{23}.

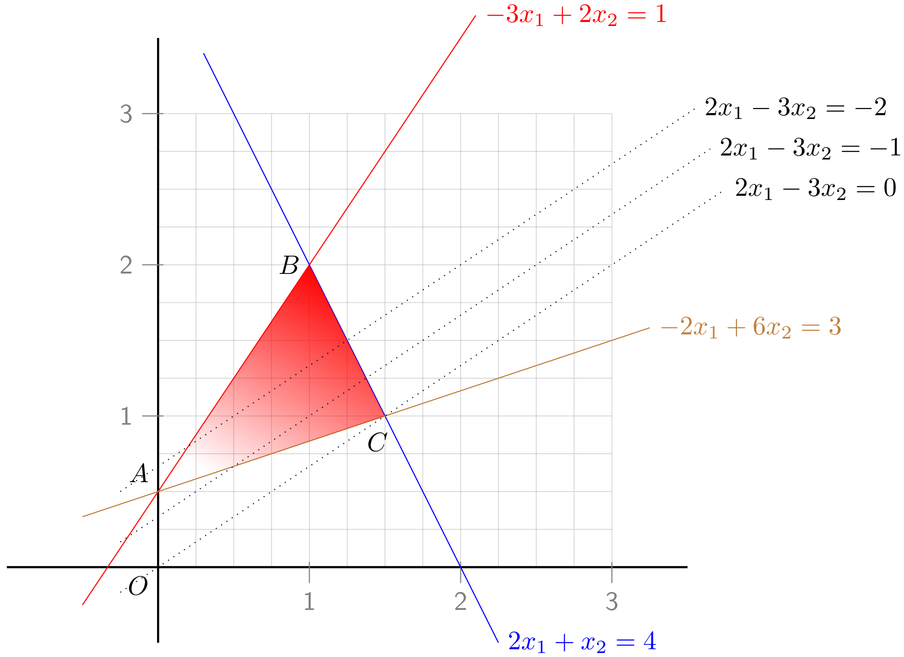
   Otra vez, es posible eliminar una tupla de R_{12} como resultado de una camino inconsistencia con x_3. Esta vez, la tupla $(3, b)$.

   $(4, c)$ Si $x_1 = 4$, no es posible entonces dar un valor a x_3. En otras palabras, las variables x_1 y x_3 ni siquiera son arco-consistentes a menos que se elimine el valor 4 de D_1. Por lo tanto, menos aún pueden ser camino consistentes y $(4, c)$ puede eliminarse de R_{12}.

   Una vez verificados todos los casos, puede concluirse entonces que x_1 y x_2 *no* son camino consistentes respecto de x_3.

Problema 4

1. La siguiente figura muestra la región factible (en rojo) que resulta de la intersección de las regiones factibles de cada restricción (cuyas fronteras están marcadas en rojo, azul y marrón).



Los puntos A, B y C se calculan como la intersección de las rectas correspondientes y resultan, por lo tanto, de la resolución de sistemas de ecuación compatibles determinados de dos variables con dos ecuaciones (cuyo cálculo se omite aquí):

$$A(0, \tfrac{1}{2})$$
$$B(1, 2)$$
$$C(\tfrac{3}{2}, 1)$$

Tal y como asegura el teorema fundamental del *simplex*, la solución óptima será un *punto extremo* de la región factible. Para ver cuál será, la misma figura muestra en negro varias curvas de isobeneficio (denominadas así porque la función objetivo es de maximización). Como se puede ver, según aumenta el valor de z, el último punto que se toca de la región factible es el punto C que será la solución buscada con un valor $z = 0$.

Una forma alternativa de calcular la solución consiste en evaluar la función objetivo en cada punto extremo:

$$z(A)=c^T A=(2 \quad -3)\begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}=-\frac{3}{2}$$

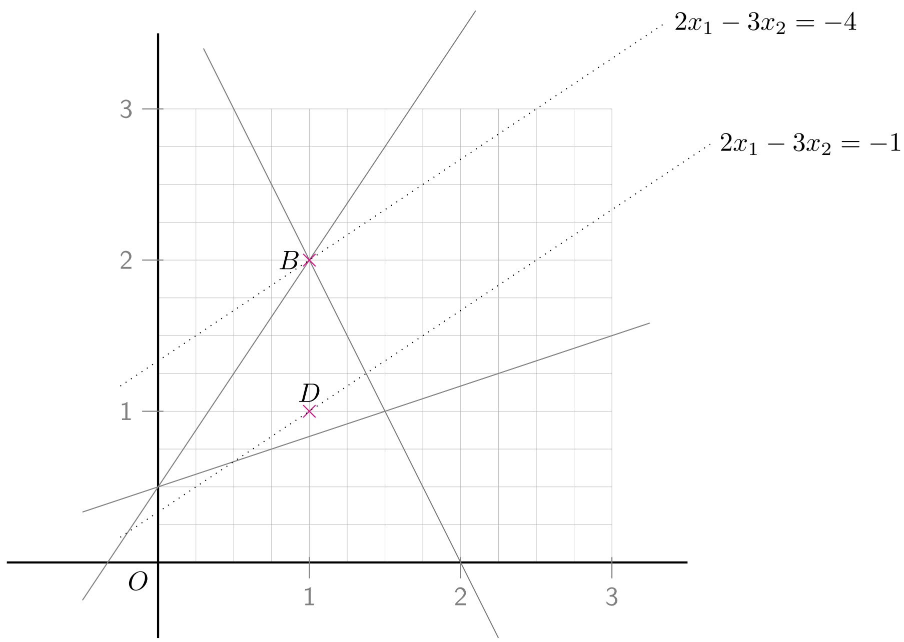
$$z(B)=c^T B=(2 \quad -3)\begin{pmatrix} 1 \\ 2 \end{pmatrix}=-4$$

$$z(C)=c^T C=(2 \quad -3)\begin{pmatrix} \frac{3}{2} \\ 1 \end{pmatrix}= 0$$

Como puede verse, el punto extremo con el mayor valor de la función objetivo es el punto C, de modo que la solución óptima del problema planteado es:

$$x^* = \begin{pmatrix} \frac{3}{2} \\ 1 \end{pmatrix}$$

2. En el segundo caso, sólo deben considerarse los puntos de la región factible que consistan en valores enteros. La siguiene figura muestra la región factible indicando los dos únicos puntos enteros que pertenenecen a élla:



Además, la misma figura muestra las curvas de isobeneficio que pasan por los puntos B y D. Como puede verse, el punto con el mayor valor de la función objetivo es el punto D y, por lo tanto, la solución óptima al problema de programación entera propuesto es:

$$x^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

3. El primer paso consiste en transformar el problema de programación lineal del enunciado en forma *estándar* de maximización.

   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 **b** no contiene términos negativos. Estará, además, 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. 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 $\geq$ están 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.

   y, en cualquier caso, las variables de holgura se añaden a la función objetivo con coeficiente nulo.

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

$$\begin{array}{rrrrrrrrrrrl} & & & & \multicolumn{8}{l}{\text{máx } z = 2x_1 - 3x_2} \\ - & 3x_1 & + & 2x_2 & + & x_3 & & & & & = & 1 \\ & 2x_1 & + & x_2 & & & + & x_4 & & & = & 4 \\ - & 2x_1 & + & 6x_2 & & & & & - & x_5 & = & 3 \\ \end{array}$$
$$\mathbf{x} \geq \mathbf{0}$$

Ahora bien, el enunciado pedía, además, transformar el problema de modo que, además, fuera posible iniciar la aplicación del algoritmo SIMPLEX con una base igual a la matriz identidad. Actualmente, los vectores columna $\mathbf{a_3}$ y $\mathbf{a_4}$ son vectores columna de la matriz identidad, pero falta aún un vector que tenga un 1 en la última posición. Para conseguirlo, simplemente se añade una *variable artificial*:

$$\text{máx } z = 2x_1 - 3x_2 - \infty x_6$$
$$\begin{array}{rrrrrrrrrrrrrl} - & 3x_1 & + & 2x_2 & + & x_3 & & & & & & & = & 1 \\ & 2x_1 & + & x_2 & & & + & x_4 & & & & & = & 4 \\ - & 2x_1 & + & 6x_2 & & & & & - & x_5 & + & x_6 & = & 3 \\ \end{array}$$
$$\mathbf{x} \geq \mathbf{0}$$

que se añade, además, a la función objetivo con una penalización infinitamente alta.

Como puede verse, la tarea de programación lineal resultante: uno, está en forma estándar de maximización; segundo, los vectores columna $\mathbf{a_3}$, $\mathbf{a_4}$ y $\mathbf{a_6}$ ya forman una base que es igual a la matriz identidad.

4. 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 tiene soluciones infinitas. Este caso se detecta cuando los denominadores y_{ij} usados en la regla de salida para la variable que entra x_i 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.
   - 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 3, tal y como se calculó ya en el apartado anterior. Por lo tanto, son variables básicas en este paso $\{x_3, x_4, x_6\}$

   $$B_0 = I_3 \qquad\qquad B_0^{-1} = I_3$$
   $$x_0^* = B_0^{-1}b = b = \begin{pmatrix} 1 \\ 4 \\ 3 \end{pmatrix} \quad z_0^* = c_{B_0}^T x_0^* = \begin{pmatrix} 0 & 0 & -\infty \end{pmatrix} \begin{pmatrix} 1 \\ 4 \\ 3 \end{pmatrix} = -3\infty$$

   b) Selección de la variable de entrada
   En las expresiones siguientes el cálculo de los vectores y_i, $y_i = B_0^{-1}a_i$, se ha embebido en el cálculo de los *costes reducidos* directamente (aunque en una iteración con una base igual a la matriz identidad, $y_i = a_i$):

$$z_1 - c_1 = c_{B_0}^T B_0^{-1} a_1 - c_1 = \begin{pmatrix} 0 & 0 & -\infty \end{pmatrix} I_3 \begin{pmatrix} -3 \\ 2 \\ -2 \end{pmatrix} - 2 = 2\infty - 2$$

$$z_2 - c_2 = c_{B_0}^T B_0^{-1} a_2 - c_2 = \begin{pmatrix} 0 & 0 & -\infty \end{pmatrix} I_3 \begin{pmatrix} 2 \\ 1 \\ 6 \end{pmatrix} + 3 \quad = -6\infty + 3$$

$$z_5 - c_5 = c_{B_0}^T B_0^{-1} a_5 - c_5 = \begin{pmatrix} 0 & 0 & -\infty \end{pmatrix} I_3 \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} - 0 = +\infty$$

En los cálculos anteriores, ∞ se ha utilizado como un símbolo cualquiera. También es posible usar una constante M muy alta (y, en particular, un valor de M mayor que la suma de los valores absolutos de todos los coeficientes en la función objetivo es suficiente para penalizar las variables artificiales). Usando ∞ como un símbolo cualquiera es posible saber qué valores son más grandes sustituyéndolo por valores arbitrariamente grandes.
En este caso particular, la variable no básica con el valor más negativo es x_2, asi que 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 (x_2) para añadirse a la base y $0 \leq j < 3$ puesto que la base tiene dimensión 3:

$$\min\left\{\frac{1}{2}, \frac{4}{1}, \frac{3}{6}\right\}$$

y hay dos cocientes con el mínimo valor, el primero y último. En tal caso, se elige arbitrariamente entre uno cualquiera de ellos y se elige x_6 por una pura cuestión de conveniencia, la de eliminar la variable artificial que siempre conlleva cálculos con el símbolo ∞.

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 $\{x_2, x_3, x_4\}$

$$B_1 = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & 1 \\ 6 & 0 & 0 \end{pmatrix} \qquad B_1^{-1} = \begin{pmatrix} 0 & 0 & \frac{1}{6} \\ 1 & 0 & -\frac{1}{3} \\ 0 & 1 & -\frac{1}{6} \end{pmatrix}$$

$$x_1^* = B_1^{-1} b = b = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{7}{2} \end{pmatrix} \quad z_1^* = c_{B_1}^T x_1^* = \begin{pmatrix} -3 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{7}{2} \end{pmatrix} = -\frac{3}{2}$$

b) Selección de la variable de entrada
En este caso conviene observar que no hay ninguna necesidad de verificar el *coste reducido* de las *variables artificiales* no básicas. El motivo es que en su cálculo se substrae su coeficiente en la función objetivo. Puesto que este coeficiente es $-\infty$, la diferencia daría valores infinitamente altos que, por lo tanto, nunca pueden ser negativos. En otras palabras, usando $-\infty$ como coeficiente de penalización de variables artificiales en la función objetivo nunca podrán volver a la base, las variables artificiales que la abandonan.

$$z_1 - c_1 = c_{B_1}^T B_1^{-1} a_1 - c_1 = \begin{pmatrix} -3 & 0 & 0 \end{pmatrix} \begin{pmatrix} -\frac{1}{3} \\ -\frac{7}{3} \\ \frac{7}{3} \end{pmatrix} - 2 = -1$$

$$z_5 - c_5 = c_{B_1}^T B_1^{-1} a_5 - c_5 = \begin{pmatrix} -3 & 0 & 0 \end{pmatrix} \begin{pmatrix} -\frac{1}{6} \\ \frac{1}{3} \\ \frac{1}{6} \end{pmatrix} - 0 = \frac{1}{2}$$

La única variable con un coste reducido negativo es x_1 que será, por lo tanto, la variable elegida para entrar en la base en la siguiente iteración.

c) Selección de la variable de salida
Nuevamene, la variable de salida se calcula en atención al mínimo cociente x_1/y_{ij} donde x_i es la variable elegida en el paso anterior para añadirse a la base (x_1) e y_{ij} son las componentes de su vector $\mathbf{y}$ también calculados en el paso anterior:

$$\min \left\{ \cancel{\frac{\frac{1}{2}}{-\frac{1}{3}}}, \cancel{\frac{0}{-\frac{7}{3}}}, \frac{\frac{7}{2}}{\frac{7}{3}} \right\}$$

Como se ve, lo dos primeros cocientes tienen denominadores negativos y, por ese motivo, se desechan. Por lo tanto, la variable que sale es la última, x_4.

Paso 2 Mejora de la solución actual (iteración #2)

a) Cálculo de las variables básicas
Las nuevas variables básicas son $\{x_1, x_2, x_3\}$

$$B_2 = \begin{pmatrix} -3 & 2 & 1 \\ 2 & 1 & 0 \\ -2 & 6 & 0 \end{pmatrix} \qquad B_2^{-1} = \begin{pmatrix} 0 & \frac{3}{7} & -\frac{1}{14} \\ 0 & \frac{1}{7} & \frac{1}{7} \\ 1 & 1 & -\frac{1}{2} \end{pmatrix}$$

$$x_2^* = B_2^{-1} b = b = \begin{pmatrix} \frac{3}{2} \\ 1 \\ \frac{7}{2} \end{pmatrix} \quad z_2^* = c_{B_2}^T x_2^* = \begin{pmatrix} 2 & -3 & 0 \end{pmatrix} \begin{pmatrix} \frac{3}{2} \\ 1 \\ \frac{7}{2} \end{pmatrix} = 0$$

b) Selección de la variable de entrada
Como en el paso anterior, se obvia el cálculo del coste reducido de la variable artificial puesto que es imposible que tome un valor negativo.

$$z_4 - c_4 = c_{B_2}^T B_2^{-1} a_4 - c_4 = \begin{pmatrix} 2 & -3 & 0 \end{pmatrix} \begin{pmatrix} \frac{3}{7} \\ \frac{1}{7} \\ 1 \end{pmatrix} - 0 \ = \frac{3}{7}$$

$$z_5 - c_5 = c_{B_2}^T B_2^{-1} a_5 - c_5 = \begin{pmatrix} 2 & -3 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{14} \\ -\frac{1}{7} \\ \frac{1}{2} \end{pmatrix} - 0 = \frac{4}{7}$$

Y como el coste reducido de todas las variables básicas es no negativo, el algoritmo SIMPLEX concluye en este paso con la siguiente solución óptima:

$$x^* = \begin{pmatrix} \frac{3}{2} \\ 1 \\ \frac{7}{2} \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad z^* = 0$$

que, como puede verse, se corresponde con el punto C calculado en el primer apartado.

5. 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 ninguna variable artificial.
   - La solución es única porque los costes reducidos de la última iteración son todos estrictamente positivos. Eso significa que cualquier cambio en la base implicaría un decremento neto en el valor de la función objetivo.
   - El valor de la función objetivo está acotado porque siempre se pudo aplicar la regla de salida con denominadores que siempre fueron estrictamente positivos.

   Interpretación de los recursos La importancia de los recursos puede estudiarse con la resolución del problema dual puesto que el valor óptimo de la variable dual i-ésima representa la contribución al crecimiento de la función objetivo por unidad de recurso i-ésimo. Sin embargo, el problema no pedía calcular la solución dual y, por lo tanto, no se ofrece ninguna interpretación relacionada con ella.

Problema 5

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 una disposición parcial de los contenedores en un cuadrado de dimension $m \times m$:

   Contenedor El contenedor i-ésimo debe estar caracterizado por su tamaño $\mathtt{l}_i$ y su posición en el cuadrado como un par $(\mathtt{x}_i, \mathtt{y}_i)$. Haciendo que la posición por defecto tome valores imposibles (por ejemplo, $\mathtt{x}_i = \mathtt{y}_i = -1$) es posible distinguir los contenedores que ya han sido dispuestos en un estado de los que están pendientes por colocarse.
   Se asume, sin pérdida de generalidad, que el tamaño de los contenedores es siempre un número entero y que, por lo tanto, su disposición se hace también en coordenadas enteras.

   Cuadrado La única información que es preciso almacenar del cuadrado mínimo que inscribe todos los contenedores es su dimensión, m que representaremos con un campo específico `m`.

   Operadores En la definición de operadores es importante describir sus precondiciones/postcondiciones y, además, el coste que tienen. En este problema hay un único operador:

Disponer Este operador recibe un contenedor que aún no ha sido colocado (esto es, con $\mathtt{x}_i = \mathtt{y}_i = -1$) y una posición (x, y) y lo dispone en el cuadrado en el lugar indicado:

Precondiciones El área del cuadrado inscrita por los puntos (x, y) y $(x + \mathtt{l}_i, y + \mathtt{l}_i)$ está completamente libre.

Postcodiciones Debe actualizarse la posición del contenedor i-ésimo haciendo $\mathtt{x}_i = x$ y $\mathtt{y}_i = y$. Además, en caso de que colocando este contenedor se exceda el tamaño del cuadrado actual, es preciso registrar el nuevo tamaño del cuadrado: $m = \min\{m, x + \mathtt{l}_i, y + \mathtt{l}_i\}$.

Coste El coste de disponer un nuevo contenedor será igual al incremento en el tamaño del cuadrado y que se calcula fácilmente como $m - \min\{x + \mathtt{l}_i, y + \mathtt{l}_i\}$.
Por lo tanto, se trata de un problema de optimización con costes diferentes que, de hecho, podrían tomar también el valor cero.

2. La profundidad d de un árbol de búsqueda se define como la profundidad *mínima* a la que se encuentra la solución.

   A partir de la definición del espacio de estados del apartado anterior, un algoritmo de búsqueda de fuerza bruta dispondría un nuevo contenedor en cada nivel y, por lo tanto, todos los nodos a profundidad N (con N el número inicial de contenedores a disponer) son nodos solución. Por lo tanto, $d = N$.

3. Tal y como se señaló en el primer apartado, en este problema hay operadores con diferentes costes. Por lo tanto, se trata de resolver un problema de *minimización* en un grafo de estados donde los operadores tienen costes arbitrarios. Los algoritmos estudiados con este propósito son fundamentalmente dos:

   Ramificación y acotación en profundidad (DFBnB) Tiene un coste de memoria lineal pero puede re-expandir muchos nodos en caso de que el dominio presente muchas *transposiciones* (como ocurre con todos los algoritmos de *el primero en profundidad*).

   Dijkstra Consiste en un algoritmo de *el mejor primero* donde la función de evaluación, $f(n)$ es, simplemente, el coste del camino desde el estado inicial hasta n, $g(n)$. Tiene un coste de memoria exponencial pero garantiza que cada vez que expande un nodo habrá encontrado la solución óptima hasta él puesto que los nodos se expanden en orden creciente de su valor de $f(n)$ —o, equivalentemente, de $g(n)$.

   En este problema en particular, si la expansión de nodos se hace de forma sistemática (esto es, considerando la disposición de cada bloque en cada posición disponible libre a lo largo y ancho del cuadrado exterior), entonces habrá un gran número de transposiciones y, por ello, el mejor algoritmo sería Dijkstra. Por el contrario, si se implementa un proceso más eficiente para la expansión de nodos (aprovechando, por ejemplo, las simetrías del cuadrado que inscribe los contenedores), entonces el número de transposiciones se reduciría significativamente concediéndole una buena oportunidad al primer algoritmo. Otro buen motivo para preferir Ramificación y Acotación en Profundidad es que todas las soluciones se disponen a profundidad N, de modo que se sabe con certeza que siempre se va a encontrar una solución —sin riesgo de caer en caminos infinitos, por ejemplo.

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) Considerar que todos los contenedores pueden superponerse unos sobre otros. De esta forma, el cuadrado mínimo que inscribe a todos sería:

   $$h_1(n) = \max_{i=1,N} \{\mathtt{l}_i\} - \mathtt{m}$$

   Nótese que es preciso restar $\mathtt{m}$ (la dimensión del cuadrado exterior en el estado n) para asegurarse que se está estimando el incremento en el tamaño del cuadrado sin sobreestimar el tamaño final.

Desgraciadamente, esta heurística no está informada en absoluto puesto que estimará que, para cualquier estado, las dimensiones finales del cuadrado serán exactamente $\max_{i=1,N}\{\mathtt{l}_i\} \times \max_{i=1,N}\{\mathtt{l}_i\}$

b) Otra posibilidad consiste en hacer la misma relajación pero sólo para un subconjunto de los contenedores.

Asumiendo que todos los contenedores del mismo tamaño $l_i \times l_i$ pueden partirse como se desee para disponerse en el menor cuadrado posible, entonces es trivial demostrar que el lado de ese cuadrado será exactamente igual a $l_i\sqrt{N_i}$.

El motivo por el que este cálculo se hace sobre todos los contenedores N_i de las mismas dimensiones $l_i \times l_i$ es para evitar hacer consideraciones sobre los huecos libres que quedan cuando se combinan (como en el problema original), contenedores de diferentes tamaños.

Pero, ¿qué pasa entonces con el resto de cuadrados de otras dimensiones? Con todos ellos se puede razonar exactamente igual

5. La selección de un algoritmo de búsqueda informada para este caso depende, como en el tercer apartado del número de transposiciones y también, naturalmente, de la dificultad de los problemas:

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.