Departamento de Informática
Universidad Carlos III de Madrid

Grado en Informática

Heurística y Optimización

27 de Junio 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)

Una empresa debe decidir entre tres candidatos para un puesto informático. El primer candidato, Roberto, demanda 75.000 € anuales, pero sus habilidades garantizan un retorno de la inversion de 125.00 € anuales. Los otros dos candidatos son Darío y Adriana, que demandan respectivamente un salario de 60.000 y 55.000 €. El retorno de cualquiera de ellos está estimado en 115.000 €.

Claramente, la empresa desea maximizar sus beneficios, calculados como la diferencia entre el retorno estimado en un candidato, y el coste de contratarlo.

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ puntos)** Modeliza el problema de elegir entre uno de los tres candidatos propuestos como un problema de grafos, **que distinga claramente entre las acciones de invertir dinero y recibir retorno**.

Indica claramente el significado y propósito de cada decisión tomada. ¿Qué significan los vértices? ¿Y los arcos? ¿El problema tiene costes?

Muestra el grafo resultante del caso propuesto en el enunciado según la modelización propuesta.

(b) **(½ puntos)** ¿Qué algoritmo de *Programación Dinámica* sugieres para resolver óptimamente este problema?

(c) **(½ puntos)** Aplica el algoritmo elegido en el apartado (b) al grafo obtenido en el apartado (a) indicando claramente todos los pasos dados, y el resultado obtenido.

Pregunta 2 (1½ puntos)

Dado un conjunto de números naturales $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, se pide determinar si existe un subconjunto $\mathcal{A}$ de $\mathcal{S}$, $\mathcal{A} \subseteq \mathcal{S}$, que contenga exactamente m elementos ($m \leq n$) de modo que no haya dos elementos s_i y s_j en el conjunto $\mathcal{A}$ que verifiquen que:

- ($s_i \mod m = s_j \mod m$), o que
- $\lfloor \frac{s_i}{m} \rfloor = \lfloor \frac{s_j}{m} \rfloor$

donde m es una constante del problema.

Por ejemplo, considérese el conjunto $\mathcal{S} = \{12, 14, 20, 87, 105, 107\}$ con 7 elementos, y se pide resolver el problema para $m = 4$. El subconjunto $\mathcal{A} = \{14, 20, 87, 105\}$ es una solución correcta, puesto que el módulo 4 de todos sus miembros es diferente, así como el resultado del cociente entero entre 4.

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ puntos)** Se pide modelar el problema *genérico* de encontrar un subconjunto $\mathcal{A}$, a partir de otro $\mathcal{S}$ y un valor fijo de m, como un problema de *satisfacción de restricciones*.

Indica claramente el significado y propósito de cada decisión tomada.

Ejemplifica la modelización sugerida con el ejemplo dado en el enunciado.

(b) **(½ puntos)** Demuestra si las variables que determinan la inclusión de los números 12 y 107 en el subconjunto $\mathcal{A}$ son *arco-consistentes* o no, ¿es necesario hacer alguna modificación para que lo sean?

(c) **(½ puntos)** Usa la modelización sugerida en el apartado (a) para probar que el subconjunto $\mathcal{A} = \{14, 20, 87, 105\}$ es, efectivamente, una solución correcta del conjunto $\mathcal{S} = \{12, 14, 20, 87, 105, 107\}$ para $m = 4$.

Pregunta 3 (3½ puntos)

Considerése el siguiente problema de Programación Lineal:

$$\begin{array}{rcrcrcr} & & \max z = 2x_1 + 3x_2 & & & & \\ 3x_1 & + & x_2 & \geq & & & 4 \\ x_1 & + & 3x_2 & \leq & - & & 4 \\ x_1 & + & x_2 & \leq & & & 2 \\ & & \mathbf{x} \geq \mathbf{0} & & & & \end{array}$$

Se pide responder razonadamente las siguientes preguntas:

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

(b) **(½ puntos)** Se dice que una restricción es *redundante* si eliminándola se tiene la misma región factible. ¿Hay alguna restricción redundante en el problema de Programación Lineal anterior? Si o no y por qué.

Considérese ahora el mismo problema de Programación Lineal en el que, sin embargo, la función objetivo es de minimización:

(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) **(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

(e) **(½ puntos)** Muestra el problema *primal* en forma *simétrica* de maximización, y el problema *dual* asociado a él.

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

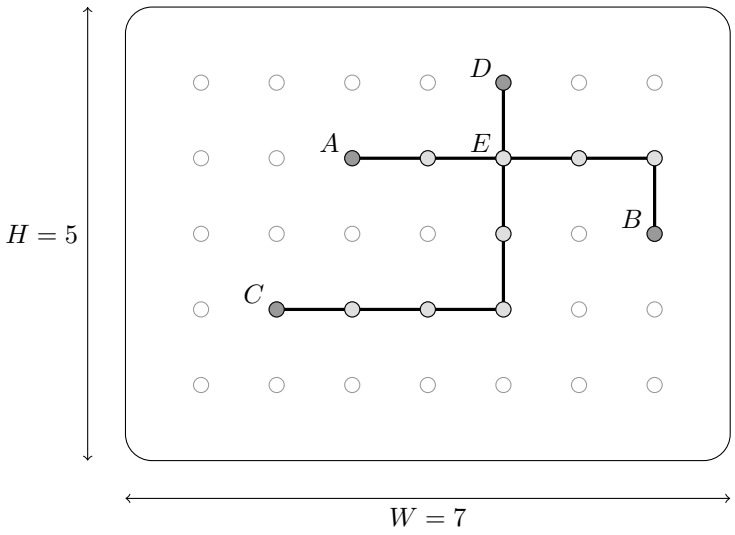
(g) **(½ puntos)** Interpretar las soluciones halladas y explicar qué conclusiones pueden extraerse.

Pregunta 4 (3½ puntos)

Una empresa de fabricación de placas madre está interesada en optimizar el diseño de las conexiones en sus placas. Las placas madre fabricadas por la empresa tienen dimensiones $W \times H$, donde W es el ancho

y H es la altura. Las placas madre disponen de posiciones marcadas con círculos entre las que pueden trazarse segmentos de silicio, con el uso de un punzón, que las conectan. El objetivo es minimizar la cantidad de silicio que se debe consumir para conectar una cantidad arbitraria de n pares $\{s_i, t_i\}_{i=1}^{n}$, habida cuenta de que los segmentos de silicio pueden cruzarse entre ellos.

La siguiente figura muestra una solución óptima para conectar el punto $A(3,4)$ con el punto $B(7,3)$, y el punto $C(2,2)$ con el punto $D(5,5)$, sobre una placa madre con un ancho igual a 7, y un alto igual a 5. Los puntos usados están marcados en gris —distinguiendo los extremos, más oscuros, de los intermedios, más claros. Nótese que el punto $E(5,4)$ es usado en ambas trayectorias.



Se pide responder razonadamente las siguientes preguntas:

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

(b) **(½ puntos)** ¿Cuál es el factor de ramificación de este espacio de estados?

(c) **(½ puntos)** Habida cuenta de que queremos encontrar soluciones óptimas, ¿qué algoritmo de búsqueda *no informada* sugerirías para su resolución? ¿Por qué?

(d) **(½ puntos)** Diseñar una función heurística $h(n)$ que sea *admisible* y que esté bien informada.

Considérese ahora que hay dos punzones disponibles para trazar simultáneamente los segmentos de silicio entre los puntos de interés:

(e) **(½ puntos)** ¿Es preciso modificar la función heurística del apartado (d) para que la función heurística siga siendo admisible? Si fuera preciso, indica las modificaciones necesarias para garantizar que siga siéndolo.

(f) **(½ puntos)** Habida cuenta que la función obtenida en el apartado (e) es admisible, ¿qué algoritmo de búsqueda *heurística* es el más indicado para resolver este problema óptimamente?

Considérese, por último, el caso de un único punzón en el que, sin embargo, en vez de conectar pares de nodos se solicita conectar una cantidad arbitraria de secuencias dadas como puntos ordenados. Por ejemplo, la figura anterior muestra una solución óptima para el caso en que las solicitudes son conectar los puntos $\{A, E, B\}$ y $\{C, E, D\}$:

(g) **(½ puntos)** ¿Es preciso modificar la función heurística del apartado (d) para que la función heurística siga siendo admisible? Si fuera preciso, indica las modificaciones necesarias para garantizar que siga siéndolo.

Soluciones del examen de Heurística y Optimización Junio 2016

Problema 1

1. El primer apartado solicita explícitamente convertir el problema de la contratación de n candidatos en un problema de grafos. En particular, se mostrará en este apartado, que el problema es *isomorfo* al del cálculo del camino más corto en un grafo en el que hay costes positivos y negativos.

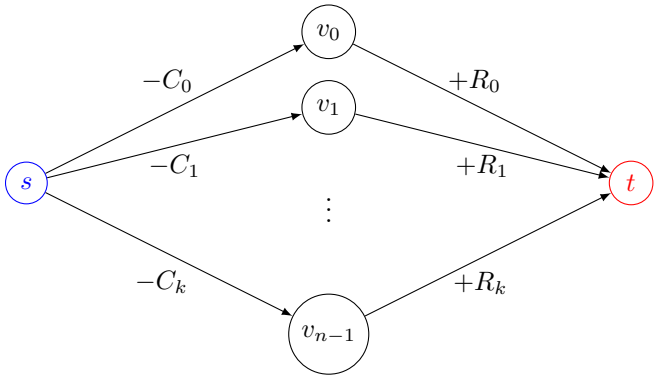
   Los vértices del grafo se dividen en tres grupos diferentes:

   - Un vértice *inicial* que representa la configuración de salida del problema, en el que no se ha contratado a ningún candidato. Se denota con la letra s, y en lo sucesivo se mostrará en color azul.
   - Hasta n vértices, denotados con la letra $v_i, 0 \leq i < n$, donde el vértice v_i representa la contratación del candidato i-ésimo.
   - Un último vértice, denotado con la letra t, que indica que ya hay contratado un candidato, y sólo uno, y que además se recibe el retorno esperado de su contratación. En los diagramas que siguen se mostrará en color rojo.

   Por lo tanto, hay dos tipos de arcos:

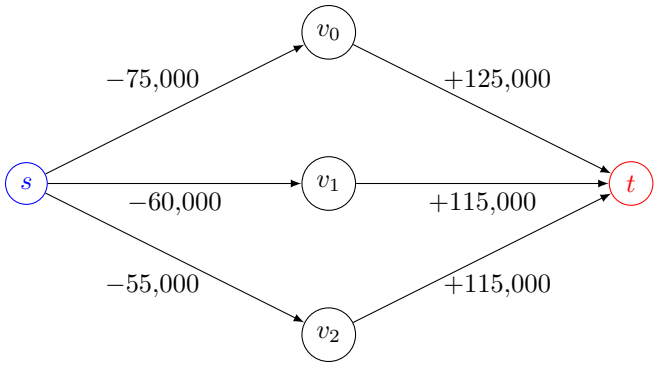
   - Del estado inicial, s a cada uno de los vértices v_i, que representan la acción de contratar al candidato i-ésimo. El coste de cada arco, por lo tanto, es exactamente igual al coste de contratarlo. Puesto que se trata de una inversión, estos costes tendrán costes negativos.
   - Desde cada vértice v_i hasta el estado final t, representando el retorno de la contratación del candidato i-ésimo. Puesto que estas acciones implican un retorno financiero para la empresa, los costes de estas acciones se representarán con costes positivos.

   La siguiente figura muestra el caso general de la modelización sugerida en este apartado:



   donde C_i representa el coste de la contratación del candidato i-ésimo, y R_i es el retorno esperado en su contratación.

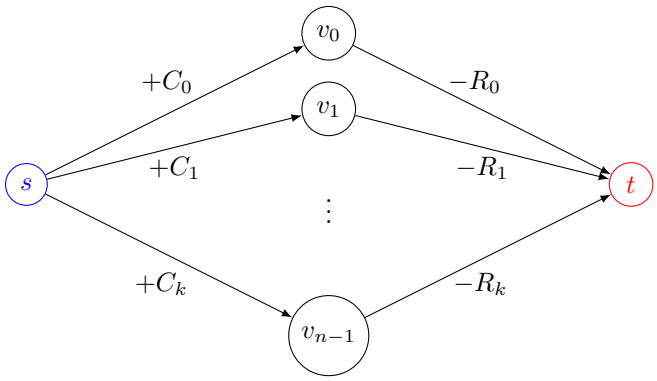
   El caso específico del problema sugerido en el enunciado es el que se muestra a continuación:



donde v_0, v_1 y v_2 representan la contratación de Roberto, Darío y Adriana, respectivamente.

2. De acuerdo con la modelización sugerida en el apartado anterior, la solución óptima es el camino de *mayor* coste desde s hasta t, puesto que ése representa el camino de mayor diferencia entre el retorno y el gasto debido a la contratación de un candidato.

   Aunque no hemos estudiado ningún algoritmo de Programación Dinámica para resolver el problema del camino más largo (que, en grafos dirigidos como éste, existe y tiene, además, una complejidad polinomial), resulta trivial convertir el modelo del apartado anterior a otro donde la solución es el camino de menor coste entre los extremos inicial, s, y final, t. Para ello, basta con cambiar el signo de todos los arcos:



   Puesto que, como se puede ver, hay tanto costes positivos como negativos, el algoritmo apropiado para resolver el problema del camino más corto en este caso es el algoritmo de Bellman-Ford-Moore.

3. El algoritmo de Bellman-Ford (o Bellman-Ford-Moore), calcula el camino óptimo desde el nodo inicial s hasta el nodo final t aplicando repetidamente la *relajación* del coste de todos los caminos, como el algoritmo de Dijkstra, pero con la diferencia de que en vez de elegir el siguiente nodo a expandir con el uso de una cola, aplica la *relajación* a todos los arcos en cada iteración, actualizando entonces el coste mínimo de alcanzar cada nodo como el mínimo del coste que tenga y el que resulta de venir del nodo origen de cada arco considerado. Haciendo esta operación una única vez se examinan todos los caminos de longitud 1. Para asegurar que se han examinado todos los caminos que existen entre el nodo inicial y el nodo final, esta operación se repite hasta $|V|-1$ veces, donde $|V|$ representa el número de vértices en el problema, que será de $(n+2)$ en este caso, donde n es el número de candidatos. Este es el motivo por el que el algoritmo de Bellman-Ford-Moore puede calcular el camino más corto incluso con costes negativos. Tiene una complejidad igual a $O(|V| \times |E|)$.

   Ahora bien, puesto que este algoritmo permite el uso de costes negativos, es preciso verificar que no existen *ciclos negativos*, esto es, caminos que parten de un vértice y llegan hasta él con un coste total estrictamente menor que cero. Si este fuera el caso, entonces la solución del problema consistiría en usar

ese ciclo indefinidamente. En el caso de este problema, sin embargo, no puede haber ciclos negativos, puesto que el modelo sugerido en el primer apartado es un grafo dirigido en el que no hay ningún bucle. Por lo tanto, no se hará esta comprobación en la ejecución que se muestra a continuación.

En el primer paso se inicializa el coste óptimo para alcanzar cada nodo u distinto del estado inicial a $+\infty$: $k[u] = +\infty, u \neq s$ donde el vector k representa las distancias óptimas desde el estado inicial. Por lo tanto, $k[s] = 0$.

A continuación, se realizarán hasta 4 iteraciones (puesto que en el grafo del problema hay 5 vértices):

Iteración #1 Se recorren todos los arcos que hay en el problema (y que son $\langle s, v_i\rangle$ y $\langle v_i, t\rangle, 0 \leq i < 3$), actualizando el coste óptimo para alcanzar cada nodo como el mínimo del valor de $k[v]$ y $k[u] + c(u, v)$, donde u es el vértice inicial, v es el vértice final del arco considerado, y $c(u, v)$ el coste del arco.

Si se recorren primero los arcos $\langle s, v_i\rangle$, entonces el valor de $k[v_i]$ será siempre igual a el mín$\{k[v_i], k[s]+c(s, v_i)\}$ = mín$\{+\infty, 0 + c(s, v_i)\}$ que, naturalmente, será siempre igual a $c(s, v_i)$. Por lo tanto, en la primera iteración: $k[v_i] = C_i$.

A continuación, se recorren los arcos $\langle v_i, t\rangle$. Si, por ejemplo, se considera primero el arco $\langle v_0, t\rangle$, entonces $k[t] =$ mín$\{k[t], k[v_0] + c(v_0, t)\} =$ mín$\{+\infty, C_0 - R_0\} = C_0 - R_0$. Este valor representa el beneficio de la compañía para la contratación del primer candidato, Roberto.

Después, se considera el arco $\langle v_1, t\rangle$ que se actualizará ahora a el mín$\{k[t], k[v_1] + c(v_1, t)\}$ = mín$\{C_0 - R_0, C_1 - R_1\}$ que, en este caso en particular será igual a $C_1 - R_1$, puesto que $60,000 - 115,000 = -55,000$ es menor que $75,000 - 125,000 = -50,000$.

Operando de nuevo de la misma manera, se compararía el mejor valor obtenido para alcanzar t con el que hay pasando por el vértice v_2, Es fácil verificar que considerando el arco $\langle v_2, t\rangle$, el coste del vértice final se actualiza a $-60,000$, que es aún menor que el valor anterior, $-55,000$.

Iteración #2 En la segunda iteración se repiten todos los cálculos exactamente igual, pero es fácil comprobar que no hay ningún cambio en los costes óptimos para alcanzar cada nodo. El algoritmo, por lo tanto, alcanza un *punto fijo*, y puede finalizar su ejecución anticipadamente.

Por último, el algoritmo debería verificar que no hay ciclos negativos pero, como ya se dijo, esta comprobación no se hará en este caso, puesto que se tiene un grafo dirigido sin bucles.

Por lo tanto, la solución óptima tiene un coste igual a -60,000, y se corresponde con el camino $\langle s, v_2, t\rangle$, que representa la contratación de Adriana.

Problema 2

1. Un problema de satisfacción de restricciones se define como una terna (X, D, C) donde $X = \{x_i\}_{i=1}^n$ es el conjunto de variables; $D = \{D_i\}_{i=1}^n$ representa los dominios de cada variable respectivamente y $C = \{C_i\}_{i=1}^m$ es el conjunto de restricciones del problema.

   En primer lugar, se sugiere una modelización sencilla pero que resultará farragosa para describir las restricciones. En esta primera modelización, si x_i representa el término elegido para entrar en el conjunto $\mathcal{A}$ en la posición i-ésima, entonces el dominio de todas las variables serán todos los números en el conjunto original, $\mathcal{S}$, $D_i = \{s_j\}_{j=1}^n, 0 \leq i < m$. Ahora bien, en este caso la restricción R_{ij} que describe la asignación legal que simultáneamente puede darse a las variables x_i y x_j es extraordinariamente complicada. De hecho, todas las restricciones son iguales y, para cualquier par de variables x_i y x_j, R_{ij} son todos los pares de números en $\mathcal{S}$ que tengan un cociente entero entre m y un módulo m diferente. Para el caso indicado en el enunciado se tiene:

$$\begin{aligned} R_{12} = \{&(12, 87), (12, 105), (12, 107),\\ &(14, 20), (14, 87), (14, 105), (14, 107),\\ &(20, 87), (20, 105), (20, 107),\\ &(87, 105)\} \end{aligned}$$

para $m = 4$. Puede comprobarse que cada tupla consiste en un par cuyos cocientes enteros entre 4 y el módulo 4 son siempre diferentes.

Ahora bien, es mucho mejor advertir que este problema es, exactamente, igual que el problema de las N-reinas en el que se ha eliminado la restricción de que dos reinas no pueden atacarse en diagonal: si se dibuja una rejilla de $m \times m$ posiciones, entonces todas las celdas en la misma fila tienen un resultado módulo m diferente, y todas las posiciones en la misma columna tienen un valor diferente del cociente entero entre m.

Por lo tanto, para su modelización se sugiere otra como la usada en el problema de las N-reinas (en vez de la presentada en primer lugar) y, en vez de tener una variable que contenga la fila que ocupa la reina en cada columna, se creará una variable que contenga el cociente entero entre m que tenga cada número con un valor módulo m específico:

$$x_i \leftrightharpoons \text{números en } \mathcal{S} \text{ con un valor módulo } m \text{ igual a } i$$

Por lo tanto, para un valor de m dado habrá exactamente m variables: $X = \{x_0, x_1, \ldots, x_{m-1}\}$

El dominio de cada variable x_i es, como se indica en la definición de la variable, todos los números dados en el problema cuyo módulo m sea exactamente igual a i: $D_i = \{s | s \in \mathcal{S} \wedge s \mod m = i\}$.

Por último, las restricciones representan los valores legales que pueden tomar simultáneamente las dos variables relacionadas por la restricción. Puesto que las restricciones relacionan a variables diferentes, la modelización escogida ya garantiza que tienen un valor diferente de módulo m y, por lo tanto, basta con comprobar la segunda y última restricción: dos números $u, v \in \mathcal{S}$ estarán en la restricción R_{ij} sí y sólo si, el primero está en el dominio D_i, el segundo en el dominio D_j (esto es, el primero tiene un módulo m igual a i, el segundo un módulo m igual a j), y el resultado del cociente entero entre m de u y v es distinto:

$$R_{ij} = \{(u, v) | u \in D_i, v \in D_j \wedge \lfloor \tfrac{u}{m} \rfloor \neq \lfloor \tfrac{v}{m} \rfloor\}$$

A continuación se muestra el resultado de aplicar la modelización propuesta al caso mostrado en el ejercicio: $\mathcal{S} = \{12, 14, 20, 87, 105, 107\}$ y $m = 4$:

a) Primero, las variables agrupan a todos los términos en $\mathcal{S}$ según el resultado del módulo 4 que, necesariamente, sólo puede tener cuatro valores (0, 1, 2 y 3). A continuación se muestran, al mismo tiempo, las variables x_i y sus dominios:

$$\begin{array}{ll} x_0 & D_0 = \{12, 20\} \\ x_1 & D_1 = \{105\} \\ x_2 & D_2 = \{14\} \\ x_3 & D_3 = \{87, 107\} \end{array}$$

Es importante advertir la importancia de esta modelización. Gracias a ella, ¡cada variable necesariamente tomará un valor módulo 4 diferente! satisfaciendo de esta manera, una restricción del problema.

b) Las restricciones, como se dijo antes, están dedicadas ahora únicamente a verificar que dos números son legales si y sólo si tienen resultados del cociente entero entre 4 distintos[1]:

$$\begin{array}{ll} R_{01} = & \{(12, 105), (20, 105)\} \\ R_{02} = & \{\cancel{(12, 14)}, (20, 14)\} \\ R_{03} = & \{(12, 87), (12, 107), (20, 87), (20, 107)\} \\ \hline R_{12} = & \{(105, 14)\} \\ R_{13} = & \{(105, 87), \cancel{(105, 107)}\} \\ \hline R_{23} = & \{(14, 87), (14, 107)\} \end{array}$$

[1] Compárense ahora estas restricciones con las obtenidas de la propuesta de modelización inicial. Mientras que aquí hay dos restricciones que sólo tienen una tupla permitida (y que, por lo tanto, sirven para iniciar la asignación consistente de valores a variables), la anterior contenía para todas las restricciones un número muy grande de tuplas permitidas.

La tabla anterior muestra para cada restricción R_{ij} el producto cruzado de todos los valores posibles en los dominios de las variables x_i y x_j. Nótese que las tuplas $(12, 14)$ y $(105, 107)$ han sido suprimidas de las restricciones R_{02} y R_{13}. El motivo es que $\lfloor \frac{12}{4} \rfloor = \lfloor \frac{14}{4} \rfloor = 3$ y que $\lfloor \frac{105}{4} \rfloor = \lfloor \frac{107}{4} \rfloor = 26$, por lo que no pueden elegirse simultáneamente.

2. Las variables que determinan la inclusión de los números 12 y 107 son x_0 y x_3. Como puede verse en la definición de R_{03} realizada en el apartado anterior:

   a) Para cada valor de la primera variable (o bien 12, o bien 20), hay siempre un valor de la segunda variable (o bien 87, o bien 107), que están soportados simultáneamente en la restricción R_{03}.

   b) Para cada valor de la segunda variable (o bien 87, o bien 107), hay siempre un valor de la primera variable (o bien 12, o bien 20), que están soportados simultáneamente en la restricción R_{03}.

   Por lo tanto, son arco consistentes.

   El mismo resultado podría haberse obtenido usando la primera modelización, aunque sería un poco más largo. Como en este caso no hay variables específicas para considerar la inclusión de un número u otro, sino la posición que ocupan, la pregunta sería tanto como verificar la arco-consistencia de que el 12 se inserte en una posición, y el 107 en otra. Si se cogen, por ejemplo, las variables x_0 y x_1 para representar la inserción de estos números en la primera y segunda posición, no sólo habría que comprobar que los cocientes enteros son distintos como antes, sino que también debe verificarse que el módulo $m = 4$ debe ser diferente. Como efectivamente lo son, también es posible verificar la arco-consistencia con aquella modelización.

3. El propósito de la programación con restricciones consiste en encontrar una asignación para cada variable de su dominio, que sea consistente con todas las restricciones del problema. Todas estas componentes ya han sido mostradas en la respuesta de la primera pregunta.

   En el caso particular que se menciona en este apartado, basta con comprobar:

   a) Todas las variables del problema tienen un valor asignado de su dominio: $x_0 = 20$, $x_1 = 105$, $x_2 = 14$ y $x_3 = 87$ y $20 \in D_0$, $105 \in D_1$, $14 \in D_2$ y $87 \in D_3$.

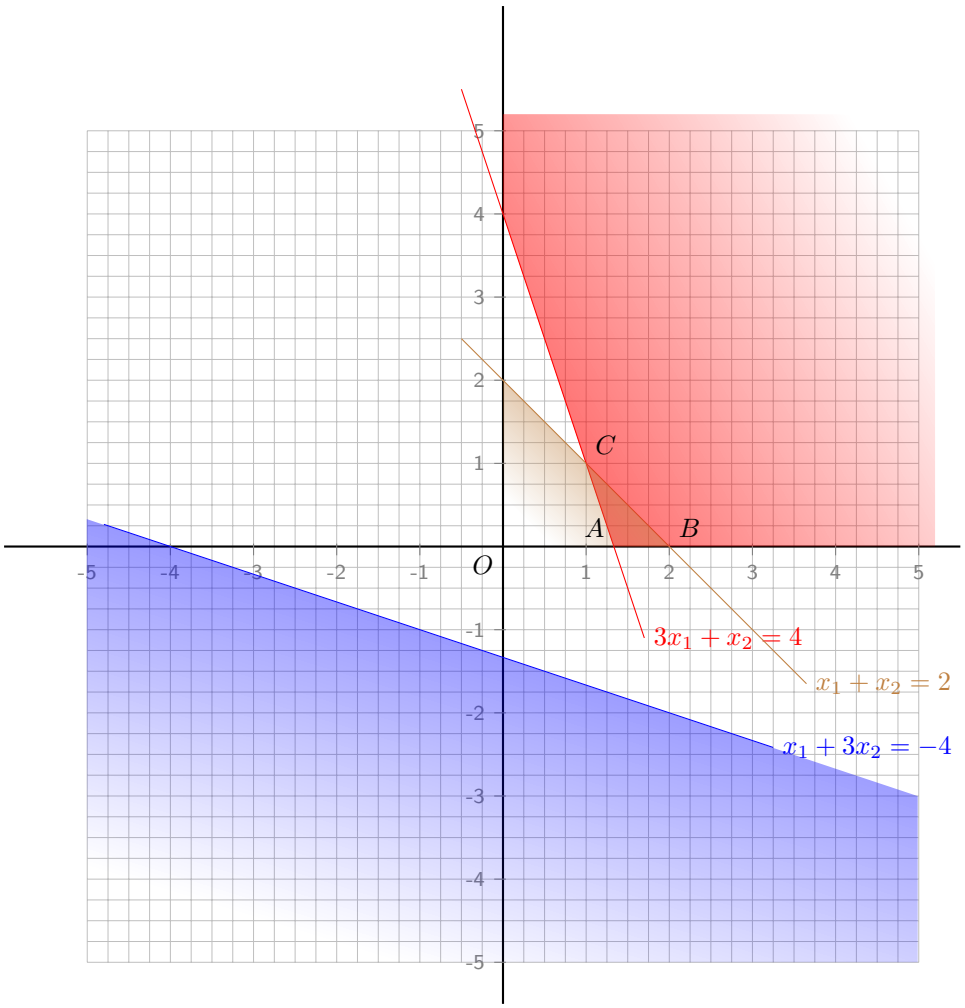
   b) Todas las asignaciones son congruentes con las restricciones del problema:

$x_0 = 20$	$x_1 = 105$	$(20, 105) \in R_{01}$
$x_0 = 20$	$x_2 = 14$	$(20, 14) \in R_{02}$
$x_0 = 20$	$x_3 = 87$	$(20, 87) \in R_{03}$
$x_1 = 105$	$x_2 = 14$	$(105, 14) \in R_{12}$
$x_1 = 105$	$x_3 = 87$	$(105, 87) \in R_{13}$
$x_2 = 14$	$x_3 = 87$	$(14, 87) \in R_{23}$

   Con lo que se prueba, como se pedía, que el conjunto $\mathcal{A} = \{14, 20, 87, 105\}$ es una solución correcta del problema para $\mathcal{S} = \{12, 14, 20, 87, 105, 107\}$ y $m = 4$.

Problema 3

1. La siguiente figura muestra las regiones factibles que se corresponden con cada restricción.

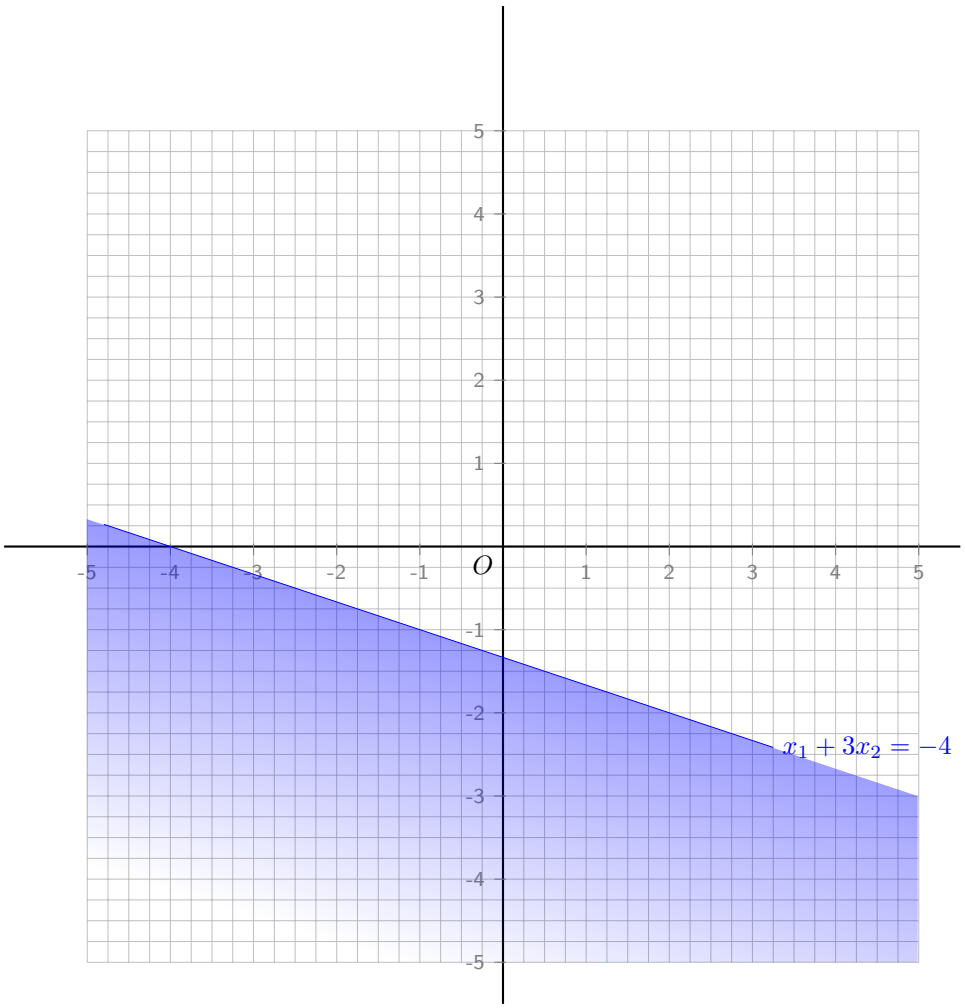


Aunque la intersección de la primera (en rojo) y tercera restricción (en marrón), crean una región convexa entre los puntos A, B y C, la segunda restricción no tiene puntos en el primer cuadrante (donde $x_1, x_2 \geq 0$)[2] y, por lo tanto, la intersección de todas las regiones es vacía. Por lo tanto, $\mathcal{F} = \varnothing$, y la tarea de Programación Lineal propuesta es infactible.

2. La pregunta, por supuesto, se refiere a alguna de las tres primeras restricciones, y no a la restricción $\mathbf{x} \geq \mathbf{0}$ que inevitablemente debe verificarse siempre. Por lo tanto, habida cuenta de que estas restricciones no pueden eliminarse, efectivamente las restricciones primera y tercera son redundantes, y podrían eliminarse sin ningún problema, obteniendo el mismo resultado, que la región factible es vacía y, por lo tanto, el problema no tiene solución.

   La siguiente gráfica muestra como, eliminando la primera y tercera restricción, la intersección de la segunda restricción con las restricciones $\mathbf{x} \geq \mathbf{0}$ produce la misma región factible, la región vacía:

[2] De hecho, la región marcada en azul no es factible, puesto que no verifica que $x_1, x_2 \geq 0$. El único propósito de marcar el área en azul, es el de enfatizar que no hay intersección con las regiones factibles de las restricciones primera y tercera.



En las secciones que siguen se propondrá, cuando aplique, la resolución con una tarea de Programación Lineal donde se eliminan las restricciones redundantes (lo que no podría alterar el resultado, puesto que se preserva la región factiblee), y donde se conserva el problema original, para demostrar que el resultado es efectivamente el mismo. Además, se verificará que las soluciones pueden interpretarse también sobre la resolución gráfica propuesta en el primer apartado.

3. A partir de este apartado se pide considerar la misma tarea de Programación Lineal, pero considerando el caso de minimización, en vez de maximización:

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

En este apartado se solicita transformar el problema de programación lineal indicado 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 $\mathbf{b}$ 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, necesita aún verificar:

- Que la función objetivo sea de la forma de maximización.
- Que el recurso de la segunda restricción sea estrictamente positivo.
- Que todas las restricciones sean de la forma de igualdad. Para ello, se añaden variables de holgura de acuerdo con las siguientes reglas:
  - 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:

$$\begin{array}{c} \text{máx } z = -2x_1 - 3x_2 \\ \begin{array}{rcrcrcrcrcl} & 3x_1 & + & x_2 & - & x_3 & & & & = & 4 \\ - & x_1 & - & 3x_2 & & & - & x_4 & & = & 4 \\ & x_1 & + & x_2 & & & & & + \; x_5 & = & 2 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

después de haber realizado las siguientes operaciones:

- La función objetivo, originalmente de minimización, es ahora de maximización simplemente multiplicándola por -1.
- La segunda restricción tiene originalmente un recurso negativo, $b_2 = -4$. Para cambiar el signo, simplemente se multiplica toda la restricción por -1 quedando: $-x_1 - 3x_2 \geq 4$.
- Las restricciones primera y, también la segunda después de modificarla, están acotadas inferiormente, de modo que para convertirlas en igualdades, se resta una variable de holgura (x_3 y x_4 respectivamente), que se añade a la función objetivo con coeficiente nulo.
- La última restricción está acotada superiormente y, por lo tanto, se convierte en una igualdad sumando una última variable de holgura, x_5, que también se añade a la función objetivo con coeficiente nulo.

Ahora bien, el enunciado solicitaba transformar el problema de Programación Lineal mostrado más arriba, de modo que fuera posible iniciar la aplicación del SIMPLEX con el uso de la matriz identidad. Se observa que sólo a_5 contiene una columna de la matriz identidad, de modo que es preciso añadir los vectores columna (1 0 0) y (0 1 0). Para ello, basta con añadir *variables artificiales*, para representar estos vectores columnas que, en la siguiente modelización aparecen como a_6 y a_7:

$$\begin{array}{c} \text{máx } z = -2x_1 - 3x_2 - \infty x_6 - \infty x_7 \\ \begin{array}{rcrcrcrcrcrcl} & 3x_1 & + & x_2 & - & x_3 & & & + \; x_6 & & & = & 4 \\ - & x_1 & - & 3x_2 & & & - & x_4 & & + \; x_7 & = & 4 \\ & x_1 & + & x_2 & & & & & & & = & 2 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

que, como se muestra, se añaden también a la función objetivo con coeficiente $-\infty$.

Otra solución correcta, habría consistido en eliminar las restricciones redundantes que, como se vió en la solución de la pregunta anterior, son la primera y tercera restricción. Por lo tanto, si se eliminan se considera el problema original siguiente:

$$\begin{array}{c} \text{mín } z = 2x_1 + 3x_2 \\ x_1 \quad + \quad 3x_2 \quad \leq \quad - \quad 4 \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

que, con la aplicación de las reglas anteriores, queda en forma estándar de maximización como sigue:

$$\begin{array}{c} \text{máx } z = -2x_1 - 3x_2 \\ -x_1 \quad - \quad 3x_2 \quad - \quad x_3 \quad = \quad 4 \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

donde x_3 (¡no confundir con la variable del mismo nombre en la forma estándar anterior!), es la variable de holgura que se resta para forzar la igualdad. Ahora, como en el caso anterior, no es posible iniciar todavía SIMPLEX con una matriz igual a la identidad (que debería tener dimensión 1, puesto que sólo hay una restricción), puesto que no existe el vector columna (1). Para ello, basta con añadir una *variable artificial*, x_4 (otra vez, ¡no confundir con la variable con el mismo nombre en la forma estándar anterior!) que se añade también a la función objetivo con coeficiente $-\infty$:

$$\begin{array}{c} \text{máx } z = -2x_1 - 3x_2 - \infty x_4 \\ -x_1 \quad - \quad 3x_2 \quad - \quad x_3 \quad + \quad x_4 \quad = \quad 4 \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

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 costes reducidos calculados en la regla de entrada tienen todos valores positivos y uno, al menos, vale cero.
   - 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.

   A continuación se procede a resolver el mismo problema de dos formas diferentes: a partir de la tarea de Programación Lineal que considera todas las restricciones; y también, resolviendo el problema que considera únicamente la restricción que queda después de eliminar las que son redundantes.

   Por lo tanto, en primer lugar se resuelve la tarea de Programación Lineal completa, con todas sus restricciones:

$$\begin{array}{c} \text{máx } z = -2x_1 - 3x_2 - \infty x_6 - \infty x_7 \\ \begin{array}{rcrcrcrcrcrcl} & 3x_1 & + & x_2 & - & x_3 & & + & x_6 & & & = & 4 \\ - & x_1 & - & 3x_2 & & - & x_4 & & & + & x_7 & = & 4 \\ & x_1 & + & x_2 & & & + & x_5 & & & & = & 2 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

Iteración 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 $\{x_6, x_7, x_5\}$ —nótese el orden, que debe ser éste para obtener exactamente la matriz identidad $I_{3\times 3}$:

$$B_0 = I_3 \qquad B_0^{-1} = I_3$$
$$x_0^* = B_0^{-1}b = b = \begin{pmatrix}4\\4\\2\end{pmatrix} \quad z_0^* = c_{B_0}^T x_0^* = \begin{pmatrix}-\infty & -\infty & 0\end{pmatrix}\begin{pmatrix}4\\4\\2\end{pmatrix} = -8\infty$$

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, $y_i = a_i$):

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

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

$$z_3 - c_3 = c_{B_0}^T B_0^{-1} a_3 - c_3 = \begin{pmatrix}-\infty & -\infty & 0\end{pmatrix} I_3 \begin{pmatrix}-1\\0\\0\end{pmatrix} - 0 = +\infty$$

$$z_4 - c_4 = c_{B_0}^T B_0^{-1} a_4 - c_4 = \begin{pmatrix}-\infty & -\infty & 0\end{pmatrix} I_3 \begin{pmatrix}0\\-1\\0\end{pmatrix} - 0 = +\infty$$

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 simplemente sustituyéndolos por valores arbitrariamente grandes.
En este caso particular, la variable no básica con el valor más negativo es x_1, 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_1) para añadirse a la base y $0 \leq j < 3$ puesto que la base tiene dimensión 3:

$$\text{mín}\left\{\frac{4}{3}, \frac{4}{\not -1}, \frac{2}{1}\right\}$$

donde, como puede verse, se descartan aquellos cocientes con denominador negativo o nulo. Puesto que el primero es el menor de todos, entonces se saca la primera variable de la base: x_6.

Iteración 1 Mejora de la solución actual

a) Cálculo de las variables básicas
Puesto que en la iteración anterior se ha determinado reemplazar la variable x_6 por la variable x_1, la nueva base de esta iteración estará formada por las variables $\{x_1, x_5, x_7\}$:

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

$$x_1^* = B_1^{-1}b = b = \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \\ \frac{16}{3} \end{pmatrix} \quad z_1^* = c_{B_1}^T x_1^* = \begin{pmatrix} -2 & 0 & -\infty \end{pmatrix} \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \\ \frac{16}{3} \end{pmatrix} = -\frac{2}{3}\left(1 + \frac{8}{3}\infty\right)$$

donde, como se puede ver, el valor de la función objetivo ha crecido, como se esperaba, de -8∞ a $-\frac{2}{3}\left(1+\frac{8}{3}\infty\right)$.

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.

$$z_2 - c_2 = c_{B_1}^T B_1^{-1} a_2 - c_2 = \begin{pmatrix} -2 & 0 & -\infty \end{pmatrix} \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \\ -\frac{8}{3} \end{pmatrix} + 3 = \frac{1}{3}(7 + 8\infty)$$

$$z_3 - c_3 = c_{B_1}^T B_1^{-1} a_3 - c_3 = \begin{pmatrix} -2 & 0 & -\infty \end{pmatrix} \begin{pmatrix} -\frac{1}{3} \\ \frac{1}{3} \\ -\frac{1}{3} \end{pmatrix} - 0 = \frac{1}{3}(2 + \infty)$$

$$z_4 - c_4 = c_{B_1}^T B_1^{-1} a_4 - c_4 = \begin{pmatrix} -2 & 0 & -\infty \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} - 0 = +\infty$$

Como quiera que todos los costes reducidos son positivos, SIMPLEX acaba en la segunda iteración, con la siguiente solución:

$$x^* = \begin{pmatrix} \frac{1}{3} \\ 0 \\ 0 \\ 0 \\ \frac{2}{3} \\ 0 \\ \frac{16}{3} \end{pmatrix} \quad z^* = -\frac{2}{3}\left(1 + \frac{8}{3}\infty\right)$$

Como puede verse, la solución óptima encontrada por SIMPLEX es única (puesto que todos los costes reducidos son estrictamente positivos para sustituciones arbitrariamente altas del símbolo $+\infty$), y consiste en un valor estrictamente positivo, para una de las variables artificiales. Por lo tanto, se trata de un problema infactible.

Ya se había mostrado, de hecho, en el primer apartado que la región factible es vacía y, por lo tanto, bien poco importa maximizar o minimizar, puesto que, con una región factible vacía, es imposible calcular mínimos o máximos.

A continuación, se resuelve el mismo problema después de eliminar las restricciones redundantes, para mostrar que había una forma mucho más sencilla de resolver el mismo problema:

$$\begin{array}{c} \text{máx } z = -2x_1 - 3x_2 - \infty x_4 \\ -x_1 - 3x_2 - x_3 + x_4 = 4 \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

Iteración 0 Cálculo de una solución factible inicial

a) Cálculo de las variables básicas
En el primer paso, la base (de dimensión 1, o sea, un escalar) estará formada, únicamente, por la variable artificial, x_4:

$$\begin{array}{cc} B_0 = I_1 & B_0^{-1} = I_1 \\ x_0^* = B_0^{-1} b = b = \left(4\right) & z_0^* = c_{B_0}^T x_0^* = \left(-\infty\right)\left(4\right) = -4\infty \end{array}$$

b) Selección de la variable de entrada

$$\begin{aligned} z_1 - c_1 &= c_{B_0}^T B_0^{-1} a_1 - c_1 = \left(-\infty\right) I_1 \left(-1\right) + 2 = +\infty + 2 \\ z_2 - c_2 &= c_{B_0}^T B_0^{-1} a_2 - c_2 = \left(-\infty\right) I_1 \left(-3\right) + 3 = 3\left(\infty + 1\right) \\ z_3 - c_3 &= c_{B_0}^T B_0^{-1} a_3 - c_3 = \left(-\infty\right) I_1 \left(-1\right) - 0 = +\infty \end{aligned}$$

Y, como ocurriera antes, la ejecución de SIMPLEX concluye en la primera iteración, puesto que todos los costes reducidos son estrictamente positivos (para cualquier sustitución del símbolo $+\infty$ por una constante arbitrariamente grande). La solución final es:

$$x^* = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 4 \end{pmatrix} \quad z^* = -4\infty$$

Y puesto que hay una variable artificial, x_4, con un valor estrictamente positivo, se prueba entonces en una sola iteración que el problema original es infactible.

5. Un problema de Programación Lineal está en forma *simétrica* de maximización, si y sólo si:

   a) La función objetivo z es de la forma de maximización
   b) Todas las restricciones son del tipo $\leq$
   c) Todas las variables de decisión son no negativas

Con el propósito de hacer esta parte más completa, se considerará ahora la tarea de Programación Lineal completa (esto es, sin eliminar las restricciones redundantes):

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

que queda, en forma simétrica de maximización como sigue:

$$\begin{array}{c} \text{máx } z = -2x_1 - 3x_2 \\ \begin{array}{rcrcr} -3x_1 & - & x_2 & \leq & -4 \\ x_1 & + & 3x_2 & \leq & -4 \\ x_1 & + & x_2 & \leq & 2 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

donde las únicas transformaciones que se han realizado han sido: de una parte, se ha multiplicado la función objetivo por -1 para convertirla en el tipo de maximización; por otra parte, se ha multiplicado también por -1 la primera restricción para que sea del tipo $\leq$.

La tarea de Programación Lineal escrita en forma simétrica como se ha indicado es, por lo tanto, el problema *primal* que se pedía en el enunciado. A partir de él, es posible convertir el problema en forma *dual* como se indica a continuación:

$$\begin{array}{c} \text{mín } w = -4x'_1 - 4x'_2 + 2x'_3 \\ \begin{array}{rcrcrcrcr} - & 3x'_1 & + & x'_2 & + & x'_3 & \geq & - & 2 \\ - & x'_1 & + & 3x'_2 & + & x'_3 & \geq & - & 3 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

6. Para la resolución de este apartado, basta con recordar la *interpretación 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 al problema dual (en forma *simétrica*) del primal —en particular, el calculado en el apartado anterior. 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 explícitamente 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.

   Ahora bien, tal y como ya se ha mostrado (tanto gráfica, como analíticamente, eliminando las restricciones redundantes y considerándolas todas), el problema es infactible. Por lo tanto, no existe una base B que sea solución del problema primal y, por ello, el problema dual también será infactible.

7. 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.

   Sin embargo, como ya se ha advertido varias veces, el problema es infactible y, por lo tanto, no hay nada más que pueda analizarse.

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 consiste en información sobre la placa madre, la demanda de puntos que deben conectarse, y la posición actual del punzón durante el trazado de segmentos entre puntos:

Punto Un concepto fundamental del problema es el de `Punto`, que se caracteriza por un par de coordenadas (x, y) que deben verificar siempre que están circunscritas al rectángulo de dimensiones $\texttt{W} \times \texttt{H}$, o sea, $0 \leq x \leq \texttt{W}, 0 \leq y \leq \texttt{H}$.

Placa madre Se caracteriza por sus dimensiones `W` y `H`. No es necesario almacenar ninguna información adicional, puesto que todos los puntos $(x, y), 0 \leq x \leq \texttt{W}, 0 \leq y \leq \texttt{H}$ pueden utilizarse para el trazado de segmentos de silicio.

Demanda Se dice que hay una `demanda` cuando se entregan dos puntos, de modo que se solicita la conexión de un punto $P(x, y)$ hasta otro $P'(x', y')$. Por lo tanto, cada demanda consiste en un par de puntos P y P', un campo booleano `satisfecho` que valdrá `true` si ya se ha encontrado una solución óptima para su conexión, y `false` en otro caso. Por último, debe haber un campo `solución` que sirva para almacenar el camino óptimo entre ambos `puntos`.

Lista Como el problema explica que el usuario entregará una cantidad arbitrariamente grande de `demandas`, se sugiere que su representación se haga con una lista.

Punzón La posición actual del punzón se almacena como un par de coordenadas (x, y). Si se quiere, estas coordenadas pueden tomar valores más allá de las dimensiones de la `placa madre` para indicar que no está sobre ella.

Operadores Este problema tiene un único operador `trazar`, que lleva silicio desde un punto (x, y) hasta otro adyacente a él horizontal o verticalmente (x', y'):

Precondiciones En primer lugar, el punto destino debe estar dentro de la rejilla: $0 \leq x' \leq \texttt{W}$, $0 \leq y' \leq \texttt{H}$. Además, el punto destino (x', y') debe ser horizontal o verticalmente adyacente al primer punto (x, y). Esta condición puede describirse algebraicamente de la siguiente manera: $|x' - x| + |y' - y| = 1$.

Postcondiciones La posición del puntero es la de la posición destino (x, y).

Coste Puesto que todos los movimientos son horizontales o verticales, y siempre de longitud uno, este es un problema de *costes unitarios*, en el que todas las instanciaciones del único operador (`trazar`) tienen el mismo coste, e igual a la unidad.

2. El factor de ramificación, b, se define como el número medio de sucesores que tiene cada nodo en el árbol de búsqueda desarrollado por un algoritmo de búsqueda. De la definición del espacio de estados sugerida en el apartado anterior, debería resultar claro que la expansión de un nodo consiste en la generación de las posiciones adyacentes a ellas. Por lo tanto, para calcular el factor de ramificación se contará el número de posiciones adyacentes a cada posición, y se calculará después la media.

   En una placa madre cualquiera, de tamaño $\texttt{W} \times \texttt{H}$ hay exactamente tres tipos de posiciones:

   Esquinas Se trata de las posiciones $(0, 0)$, $(0, \texttt{H} - 1)$, $(\texttt{W} - 1, 0)$ y $(\texttt{W} - 1, \texttt{H} - 1)$.

   Hay, por lo tanto, exactamente cuatro esquinas, cada una de las cuales tiene, únicamente, dos sucesores.

   Laterales Se trata de todas las posiciones que hay en los lados después de excluir las esquinas. Se trata de las posiciones: $(0, y), (\texttt{W} - 1, y), 1 \leq y < \texttt{H} - 1$, y $(x, 0), (x, \texttt{H} - 1), 1 \leq x < \texttt{W} - 1$.

   Hay, $2(\texttt{W} - 2 + \texttt{H} - 2) = 2(\texttt{W} + \texttt{H} - 4)$ posiciones, cada una de las cuales tiene exactamente tres sucesores —el 4 que se resta en el interior del producto, se corresponde exactamente con las cuatro esquinas calculadas en el apartado enterior.

   Centrales Son el resto de las posiciones. Esto es, cualquier posición (x, y), tal que $1 \leq x < \texttt{W} - 1$ y $1 \leq y < \texttt{H} - 1$ es una posición central.

   Por lo tanto, hay exactamente $(\texttt{W} - 2) \times (\texttt{H} - 2)$ posiciones centrales, cada una de las cuales tiene exactamente cuatro sucesores.

Como en total hay `W` $\times$ `H` posiciones en la placa madre, entonces el factor de ramificación se calcula con precisión como la media aritmética que se muestra a continuación:

$$\begin{aligned} b &= \frac{1}{W \times H} \left(4 \times 2 + 2(W + H - 4) \times 3 + (W - 2)(H - 2) \times 4\right) \\ &= \frac{2}{W \times H} \left(4 + (W + H - 4) \times 3 + (W - 2)(H - 2) \times 2\right) \end{aligned}$$

En el caso particular del enunciado, donde $W = 7$ y $H = 5$:

$$\begin{aligned} b &= \frac{2}{7 \times 5} \left(4 + (7 + 5 - 4) \times 3 + (7 - 2)(5 - 2) \times 2\right) \\ &= \frac{2}{35} \left(4 + 8 \times 3 + 15 \times 2\right) \\ &= \frac{2}{35} \times 58 = 3,31 \end{aligned}$$

Resulta fácil comprobar que si la placa madre crece arbitrariamente en una cualquiera de sus dimensiones, entonces el factor de ramificación se acerca asintóticamente a 4:

$$\begin{aligned} \lim_{W \to +\infty} b &= 4 \\ \lim_{H \to +\infty} b &= 4 \end{aligned}$$

3. Puesto que se trata de un problema de *costes unitarios* (ver la solución al primer apartado), se centra entonces la atención en los algoritmos: *Primero en Profundidad*, *Primero en Amplitud* y *Primero en Profundización Iterativa*. Esto no significa que los algoritmos dedicados a encontrar soluciones óptimas con *costes arbitrarios* (*Dijkstra* y *Ramificación y Acotación en Profundidad*) no sirvan, pero o bien se convierten en algunos de los anteriores, o tienen las mismas dificultades que aquellos:

   - En términos generales, el algoritmo de el primero en profundidad no es ni *completo* (esto es, no garantiza que vaya a encontrar soluciones), ni *admisible* (de modo que si encuentra soluciones, no garantiza que sean óptimas). En este caso, puesto que el espacio de estados es finito, sí puede garantizarse que eventualmente se encontrará una solución (dado que la profundidad máxima sea lo suficientemente grande), pero no puede garantizarse en absoluto que las soluciones sean óptimas. Por lo tanto es descartado
   - El algoritmo del primero en amplitud es completo y, ademas, admisible. Ahora bien, encuentra soluciones óptimas en la profundidad de la solución. Puesto que éste es un problema de costes unitarios, entonces el coste de una solucioń es igual al de la profundidad en la que se encuentra.

     Este algoritmo, sin embargo, tienen un consumo de memoria exponencial, aunque es particularmente bueno en el caso de que haya transposiciones.
   - El algoritmo de el primero en profundización iterativa realiza varias búsquedas de el primero en profundidad secuencialmente, incrementando la profundidad máxima en cada iteración. Por lo tanto, se trata de un algoritmo completo. Además, si el incremento de la profundidad máxima entre iteraciones es igual a la unidad, entonces la soluciones encontradas serán necesariamente óptimas en la profundidad de las soluciones —que, como ya se ha mencionado, resultan ser iguales al coste de las soluciones.

     Aunque este algoritmo tiene un consumo de memoria que es lineal en la profundidad a la que se encuentra la solución óptima, tiene dificultades en los espacios de estados con transposiciones, puesto que recorre todos los caminos que pasan por cada transposición.

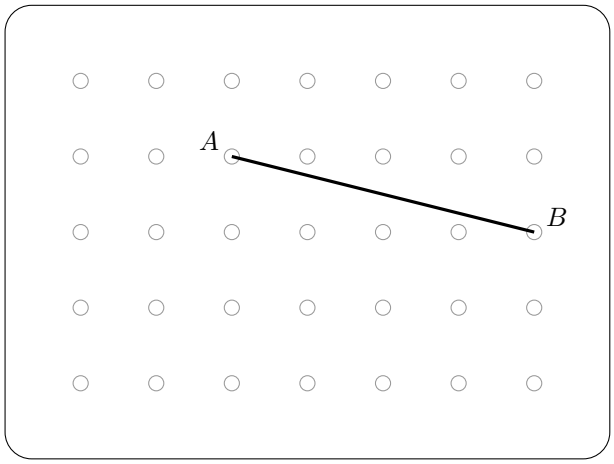
   Puesto que este problema tiene una cantidad alta de transposiciones y, además, es lógico asumir que las placas madres tienen una cantidad limitada, más bien pequeña de posiciones legales, el algoritmo recomendado es el algoritmo de el primero en amplitud.

4. Tal y como se explica en el enunciado, se trata de encontrar el camino óptimo entre todos los pares de puntos que se especifican en cada `demanda`. Como, además, es posible reutilizar puntos de la placa madre para conectar cada par de puntos, entonces cada `demanda` es un problema *independiente*.

   Consecuentemente, cualquier función heurística que sirva para resolver óptimamente la conexión de cada par de puntos, servirá para resolver el problema entero que, de hecho, se resuelve con la ejecución de un algoritmo de búsqueda heurística por cada par de nodos que deben conectarse.

   Para obtener una función heurística, se sugiere el uso 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 (en las que se considera, en los ejemplos mostrados, el primer par (A, B)):

   a) Si se relaja la restricción de adyacencia horizontal o vertical completamente, entonces sería posible trazar un segmento desde cualquier posición de la placa madre, hasta cualquier posición como se indica en la siguiente figura:

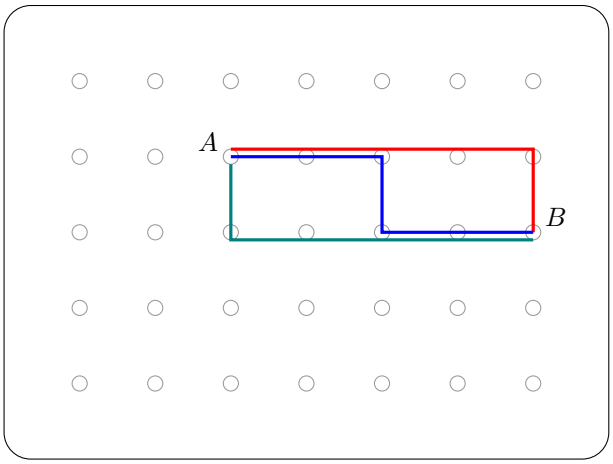
   En este caso, la cantidad de silicio mínima necesaria para conectar dos puntos $P(x, y)$ y $P'(x, y)$ sería una heurística admisible del problema original, para conectar precisamente esos dos mismos puntos:

   $$h_1 = \sqrt{(x - x')^2 + (y - y')^2}$$

   y que se conoce como *distancia euclídea* o *distancia aérea*.

   b) Sin embargo, no hace falta relajar ninguna restricción para calcular la cantidad mínima de silicio que hace falta para conectar dos puntos $P(x, y)$ y $P'(x', y')$.

   La siguiente figura, muestra varias soluciones óptimas para la conexión de los puntos A y B en colores diferentes:



En todas ellas, la cantidad óptima de silicio requerida para conectar ambos puntos es igual a la suma de la diferencia de filas y columnas entre la posición original y la posición final:

$$h_2 = |x - x'| + |y - y'|$$

conocida como la *distancia de Manhattan*.

Otra restricción que puede relajarse es que la posición adyacente esté circunscrita a la placa, y que entonces se permita el trazado de segmentos fuera de la placa. Sin embargo, no parece que esa relajación produzca estimaciones informadas para resolver el problema original.

Como la distancia de *Manhattan* es necesariamente mayor o igual (en otras palabras, *más informada*) que la distancia *euclídea*, se elige h_2 para resolver la conexión de cada par de puntos. Con esa heurística es posible resolver el problema original, resolviendo independientemente cada par de nodos.

5. No, no es necesario hacer ninguna modificación. Tal y como se indicó en el apartado anterior, todos los problemas son independientes y, por lo tanto, el número de punzones disponibles no juega ningún papel relevante en el cálculo de las soluciones óptimas —y, por lo tanto, del cálculo de la cantidad mínima de silicio que debe usarse.

   La única modificación, que no afecta a la optimalidad de las soluciones, es que mientras que en el apartado anterior se sugería resolver el problema con la aplicación de una cantidad arbitraria de algoritmos de búsqueda *secuencialmente*, la disponibilidad de una cantidad arbitraria de punzones, habilita la aplicación del mismo número de algoritmos de búsqueda heurística en *paralelo*.

6. 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. 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 . 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 primero es el más apropiado si el problema que se pretende resolver tiene muchas transposiciones, como es el caso de este problema.

7. Otra vez, la respuesta es que no. La única diferencia es que ahora, cada algoritmo de búsqueda (en secuencia, si hay un solo punzón, o en paralelo, si hay varios punzones disponibles), debe resolver el problema de encontrar la solución óptima desde un punto hasta el siguiente en el camino que se indica en cada `demanda`.

   Para ello, la función heurística h_2 obtenida en la solución del cuarto apartado sigue sirviendo para encontrar soluciones óptimas.