Departamento de Informática
Universidad Carlos III de Madrid

Grado en Informática

Heurística y Optimización

Julio 2012

Normas generales del examen

- El tiempo para realizar el examen es de **4 horas**
- Cada pregunta debe responderse en páginas separadas
- No se responderá a ninguna pregunta sobre el examen
- Si se sale del aula, no se podrá volver a entrar durante el examen
- No se puede presentar el examen escrito a lápiz

Problema 1. (5 puntos)

Considera el siguiente problema de Programación Lineal:

$$\begin{array}{rrcrcr} & \min z = 4x_1 & + & x_2 & & \\ & 2x_1 & + & x_2 & \geq & 18 \\ - & x_1 & + & x_2 & \geq & 8 \\ & 5x_1 & + & 2x_2 & \leq & -10 \\ & x_1, x_2 \geq 0 & & & & \end{array}$$

y responde razonadamente las siguientes preguntas:

1. **(0,5 puntos)** Expresar el problema en forma *canónica*
2. **(0,5 puntos)** Representa gráficamente las restricciones obtenidas en el apartado anterior indicando las curvas de isocoste
3. **(0,5 puntos)** Resuelve gráficamente el problema anterior

Considerando ahora, en su lugar, el siguiente problema de Programación Lineal:

$$\begin{array}{rrcrcr} & \max z = 2x_1 & + & x_2 & & \\ & 2x_1 & + & x_2 & \leq & 18 \\ - & x_1 & + & x_2 & \leq & 8 \\ & 5x_1 & + & 2x_2 & \geq & 0 \\ & \mathbf{x} \geq \mathbf{0} & & & & \end{array}$$

se pide responder razonadamente las siguientes cuestiones:

4. **(1 punto)** Resuelve el problema de Programación Lineal anterior con el algoritmo Simplex. Indica claramente, en cada iteración, las variables escogidas en la base y su valor así como el valor de la función objetivo z en cada paso.
5. **(0,5 puntos)** Discute el resultado indicando claramente su significado.

6. **(0,5 puntos)** ¿Se habría obtenido el mismo resultado si, en su lugar, la función objetivo hubiera sido de *minimización*? Si o no y por qué.
7. **(0,5 puntos)** ¿Cuál es el problema dual de este problema de Programación Lineal?
8. **(0,5 puntos)** Resuelve el problema dual del problema de Programación Lineal mostrado.
9. **(0,5 puntos)** ¿Cuál es la contribución por unidad del recurso que se corresponde con la primera restricción al valor óptimo de la función de coste?

Problema 2. (3 puntos)

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

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

Se pide responder razonadamente las siguientes cuestiones:

1. **(0,5 puntos)** Indica qué literales son puros y resuelve la *resolución* de la fórmula F respecto de esos literales.
2. **(1 punto)** Resolver la fórmula resultante del apartado 1 con el algoritmo de Davis-Putnam. Considera, para ello, las variables en orden ascendente de su subíndice.
3. **(0,5 puntos)** ¿Cuántas soluciones posibles hay al problema de satisfabilidad anterior a partir de los resultados del algoritmo de Davis-Putnam?
4. **(1 punto)** Resuelve el mismo problema con el algoritmo de Davis-Putnam-Logemann-Loveland. En particular, indica una ordenación *óptima* de la asignación de valor a las variables que sirva para encontrar la solución rápidamente.

Problema 3. (2 puntos)

El diseño de placas integradas se hace siempre con brazos automáticos que unen una cantidad arbitraria de puntos de contacto especificados sobre una tarjeta. Dichos puntos de contacto se especifican, normalmente, como un par de coordenadas (x, y) inscritas en el rectángulo de $N \times M$ que representa a la tarjeta. Por su parte, el brazo automático puede sobrevolar la placa hasta cualquier punto (x, y). Una vez que lo encuentra, se detiene y hace descender la punta hasta que toca la tarjeta. A partir de ese momento, puede trazar líneas (horizontales y/o verticales) que habilitan el movimiento de electrones entre puntos, poniéndolos entonces en contacto entre sí, gracias al uso de una materia específica para éllo. A partir del momento en el que el brazo desciende, debe poner todos los puntos en contacto, sin levantarse de la placa nuevamente. Durante el trazado desde un punto a otro, la cabeza sigue aplicando el material conductor incluso si está recorriendo un tramo donde ya hubiera.

Una compañía desea minimizar el consumo del material empleado para establecer las conexiones entre puntos. Se pide:

1. **(0,5 puntos)** ¿Qué algoritmo de fuerza bruta debería usarse para resolver este problema óptimamente?
2. **(0,5 puntos)** Definir una función heurística $h_1(n)$ que sea *admisible* e *informada*, explicando claramente su construcción, para la minimización del material empleado en los trazos que unen todos los puntos

Un tipo particular de cabeza, puede moverse, además, en diagonal una vez que ha descendido sobre la tarjeta. Para este caso particular se pide:

3. **(0,5 puntos)** Definir una función heurística $h_2(n)$ que sea *admisible* e *informada*, explicando claramente su construcción, para la minimización del material empleado en los trazos que unen todos los puntos en este caso particular

Actualmente, la compañía está considerando acelerar el proceso de construcción de tarjetas empleando hasta dos cabezas al mismo tiempo sobre la misma tarjeta. Las cabezas a usar permitirían movimientos horizontales y/o verticales exclusivamente. Por supuesto, cualquier cabeza podría pasar sobre puntos de contacto visitados por la otra. En este caso, si cualquier cabeza pasa por encima de un tramo donde ya hubiera material conductor, entonces no lo aplica.

4. **(0,5 puntos)** ¿Es posible emplear alguna de las funciones heurísticas definidas en los apartados 2 y 3? Si o no y por qué

Soluciones del examen de Heurística y Optimización
Julio 2012

Solución al problema 1

1. La forma canónica de un problema de Programación Lineal se caracteriza porque:

   - El objetivo es de maximización
   - Todas las restricciones son desigualdades del tipo $\leq$
   - Todas las variables de decisión son no negativas

   Por lo tanto, para convertir a la foma canónica el problema de Programación Lineal:

   $$\begin{array}{rrcrcr} & \text{mín}\, z = 4x_1 + x_2 & & & & \\ & 2x_1 & + & x_2 & \geq & 18 \\ - & x_1 & + & x_2 & \geq & 8 \\ & 5x_1 & + & 2x_2 & \leq & -10 \\ & x_1, x_2 \geq 0 & & & & \end{array}$$

   se deben aplicar las siguientes transformaciones:

   - Cambiar el sentido de la función objetivo de minimización a maximización.
     Para ello, basta con multiplicar por -1 la función z resultando $z = -4x_1 - x_2$
   - Transformar aquellas desigualdades del tipo $\geq$ en desigualdades del tipo $\leq$
     Como antes, basta con multiplicar las restricciones a transformar por -1. Las restricciones involucradas son la primera y segunda que ahora quedan entonces de la manera:

     $$\begin{array}{rrcrcr} - & 2x_1 & - & x_2 & \leq & -18 \\ & x_1 & - & x_2 & \leq & -8 \end{array}$$

   - Por último, las variables de decisión deben ser no negativas, tal y como era el caso en el problema originalmente.

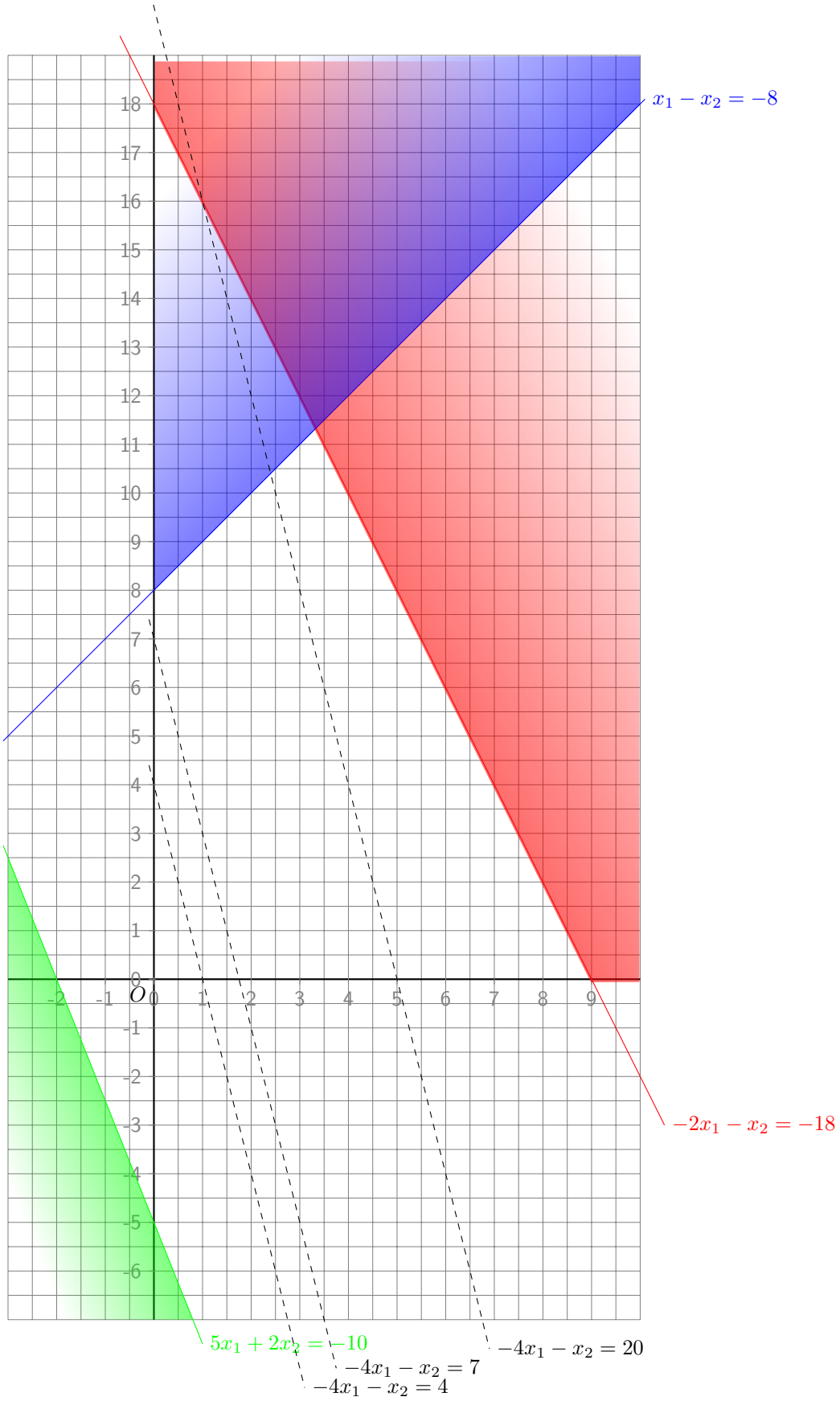
   Por lo tanto, el problema anterior expresado ahora en forma canónica es:

   $$\begin{array}{rrcrcr} & \text{máx}\, z = -4x_1 - x_2 & & & & \\ - & 2x_1 & - & x_2 & \leq & -18 \\ & x_1 & - & x_2 & \leq & -8 \\ & 5x_1 & + & 2x_2 & \leq & -10 \\ & x_1, x_2 \geq 0 & & & & \end{array}$$

2. Puesto que sólo hay dos variables de decisión, x_1 y x_2, las restricciones se pueden dibujar sobre un plano bidimensional. Obviamente, las rectas resultantes son iguales, tanto en la forma canónica como en la forma original del problema.

   La siguiente figura muestra los semiplanos que verifican cada una de las restricciones con un degradado de color. Además, las curvas de isocoste se han mostrado, para diferentes valores de la función de coste con líneas discontinuas.

   En particular, se ha tenido un cuidado especial en dibujar las intersecciones de los semiplanos de las restricciones primera y segunda con el plano $\mathbf{x} \geq \mathbf{0}$, mientras que esto no es posible con la tercera restricción.



3. Como puede verse en la figura anterior, las restricciones primera y segunda sí coinciden, pero no existe ninguna región factible. Esto es, no existe ninguna región en el semiplano positivo $\mathbf{x} \geq \mathbf{0}$ donde intersecan todos los

semiplanos factibles de las tres restricciones. En particular, tal y como se indicaba en el apartado anterior, ni siquiera hay ninguna intersección entre la región factible de la tercera restricción y el semiplano positivo $\mathbf{x} \geq \mathbf{0}$.

Por lo tanto, el problema es infactible.

4. En el cuarto apartado se pedía resolver el siguiente problema de Programación Lineal con el algoritmo Simplex:

$$\begin{array}{rrcrcl} & \max z = 2x_1 & + & x_2 & & \\ & 2x_1 & + & x_2 & \leq & 18 \\ - & x_1 & + & x_2 & \leq & 8 \\ & 5x_1 & + & 2x_2 & \geq & 0 \\ & & & \mathbf{x} \geq \mathbf{0} & & \end{array}$$

Para ello, el primer paso consiste en expresarlo en forma estándar que quedaría como sigue:

$$\begin{array}{rrcrcrcrcl} & \max z = 2x_1 & + & x_2 & & & & & & \\ & 2x_1 & + & x_2 & + & x_3 & & & = & 18 \\ - & x_1 & + & x_2 & & & + & x_4 & = & 8 \\ & 5x_1 & + & 2x_2 & & & - & x_5 & = & 0 \\ & & & & & \mathbf{x} \geq \mathbf{0} & & & & \end{array}$$

donde, como se ve, 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. Nótese que en la última restricción la variable de holgura resta puesto que originalmente consistía en una desigualdad $\geq$, mientras que las otras suman.

A continuación, se iniciará la aplicación del Simplex con una base factible inicial de dimensión 3 (puesto que el menor rango de la matriz de coeficientes es el de filas, 3), que es la que resulta de considerar en la base las variables de holgura x_3, x_4 y x_5. Con la selección de esa base, se aplicarán repetidamente las reglas de entrada y salida hasta que no sea posible mejorar el valor de la función objetivo.

Paso 0. Cálculo de una solución factible inicial

a) Cálculo de las variables básicas
Originalmente se consideran las variables x_3, x_4 y x_5:

$$B_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \quad B_0^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix},$$

Con esta base, los valores de las variables básicas se calculan como sigue:

$$\mathbf{x_0^*} = B_0^{-1} b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 18 \\ 8 \\ 0 \end{pmatrix} = \begin{pmatrix} 18 \\ 8 \\ 0 \end{pmatrix}$$

para los que la función objetivo tiene un valor igual a:

$$z_0 = c_{B_0}^T \mathbf{x_0^*} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 18 \\ 8 \\ 0 \end{pmatrix} = 0$$

donde $c_{B_0}^T$ denota los coeficientes del vector de costes **de las variables básicas** que, puesto que ahora son las variables de holgura, aparecen en la función de coste con coeficiente nulo.

b) Selección de la variable de entrada
Como en este paso, $c_{B_0}^T$ es el vector nulo, no hay ninguna necesidad de calcular los vectores $\mathbf{y}$ de las variables no básicas para obtener los costes reducidos. Sin embargo, se indican a continuación únicamente por completitud:

$$y_1 = B_0^{-1}a_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 2 \\ -1 \\ 5 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ -5 \end{pmatrix}$$

$$y_2 = B_0^{-1}a_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -2 \end{pmatrix}$$

A partir de ellos, es posible calcular los *costes reducidos* de la siguiente manera:

$$(z_1 - c_1) = c_{B_0}^T y_1 - c_1 = (0 \quad 0 \quad 0)\, y_1 - 2 = -2$$
$$(z_2 - c_2) = c_{B_0}^T y_2 - c_2 = (0 \quad 0 \quad 0)\, y_2 - 1 = -1$$

De acuerdo con la regla de la variable de entrada, lo hará aquella más negativa, que en este caso es x_1.

c) Selección de la variable de salida
Según la regla de selección de salida, se escoge aquella variable con el mínimo coeficiente x/y con denominadores estrictamente positivos[1]:

$$\theta = \text{mín}\left\{\frac{18}{2}, \frac{8}{\not{-1}}, \frac{0}{\not{-5}}\right\} = 9$$

que se corresponde con la primera variable básica. Por lo tanto, sale x_3.

Paso 1 Mejora de la solución (iteración 1)

a) Cálculo de las variables básicas
En este paso, la base está formada por las variables x_1, x_4 y x_5. Por lo tanto, la base (y su inversa) son:

$$B_1 = \begin{pmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 5 & 0 & -1 \end{pmatrix}, \quad B_1^{-1} = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{5}{2} & 0 & -1 \end{pmatrix},$$

de modo que el punto considerado en esta iteración es:

$$\mathbf{x_1^*} = B_1^{-1}b = \begin{pmatrix} 9 \\ 17 \\ 45 \end{pmatrix}$$

para el que la función objetivo tiene un valor igual a:

$$z_1 = c_{B_1}^T \mathbf{x_1^*} = (2 \quad 0 \quad 0) \begin{pmatrix} 9 \\ 17 \\ 45 \end{pmatrix} = 18$$

b) Selección de la variable de entrada
En primer lugar, se presenta el cálculo de los vectores $\mathbf{y}$:

$$y_2 = B_1^{-1}a_2 = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{5}{2} & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{3}{2} \\ \frac{1}{2} \end{pmatrix}$$

[1] Nótese que en la expresión siguiente se eliminan, en particular, aquellos cocientes con denominador negativo

$$y_3 = B_1^{-1} a_3 = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{5}{2} & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{5}{2} \end{pmatrix}$$

A partir de estos vectores, ya es posible calcular los *costes reducidos* de las variables no básicas, tal y como se indica a continuación:

$$(z_2 - c_2) = c_{B_1}^T y_2 - c_2 = (2 \quad 0 \quad 0) \begin{pmatrix} \frac{1}{2} \\ \frac{3}{2} \\ \frac{1}{2} \end{pmatrix} - 1 = 0$$

$$(z_3 - c_3) = c_{B_1}^T y_3 - c_3 = (2 \quad 0 \quad 0) \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{5}{2} \end{pmatrix} - 0 = 1$$

Como no existe ningún coste reducido negativo, el algoritmo SIMPLEX termina en este paso en el punto con $x_1 = 9$ (primera componente de $\mathbf{x_1^*}$) y $x_2 = 0$ —puesto que es una variable no básica. El valor óptimo de la función objetivo se calculó al principio de esta iteración y es $\mathbf{z}^* = 18$.

5. El algoritmo SIMPLEX terminó con un coste reducido nulo lo que indica la existencia de *soluciones infinitas*. Para conocer la recta que representa todas las soluciones podría continuarse con la aplicación del SIMPLEX un paso más para ver cuál es el siguiente extremo óptimo.

Paso 1 Mejora de la solución (Iteración 1) – cont.

c) Selección de la variable de salida
Como antes, la variable que salga será aquella que minimice el cociente siguiente:

$$\theta = \text{mín} \left\{ \frac{9}{\frac{1}{2}}, \frac{17}{\frac{3}{2}}, \frac{45}{\frac{1}{2}} \right\} = \frac{34}{3}$$

que se corresponde con la variable x_4

Paso 2 Mejora de la solución (Iteración 2)

a) Cálculo de las variables básicas
En este paso, la base está formada por las variables x_1, x_2 y x_5. Por lo tanto, la base (y su inversa) son:

$$B_2 = \begin{pmatrix} 2 & 1 & 0 \\ -1 & 1 & 0 \\ 5 & 2 & -1 \end{pmatrix}, \quad B_2^{-1} = \begin{pmatrix} \frac{1}{3} & -\frac{1}{3} & 0 \\ \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{7}{3} & -\frac{1}{3} & -1 \end{pmatrix},$$

de modo que el punto considerado en esta iteración es:

$$\mathbf{x_2^*} = B_2^{-1} b = \begin{pmatrix} \frac{10}{3} \\ \frac{34}{3} \\ \frac{118}{3} \end{pmatrix}$$

para el que la función objetivo tiene un valor igual a:

$$z_2 = c_{B_2}^T \mathbf{x_2^*} = (2 \quad 1 \quad 0) \begin{pmatrix} \frac{10}{3} \\ \frac{34}{3} \\ \frac{118}{3} \end{pmatrix} = 18$$

Obviamente, que el valor de la función objetivo no haya cambiado no debería ser una sorpresa puesto que ya sabíamos que hay soluciones infinitas. En esta iteración se pretendía conocer únicamente el segundo extremo de la recta que las representa, de modo que ahora se sabe que son todos los puntos entre $\mathbf{x_1^*} = (9 \quad 0)$ y $\mathbf{x_2^*} = (\frac{10}{3} \quad \frac{34}{3})$

Calculando ahora los costes reducidos se puede verificar que el menor de todos ellos sería nuevamente nulo.

En cualquier caso, no hacía ninguna falta calcular el SIMPLEX para observar que efectivamente el problema tiene soluciones infinitas. En particular, la función objetivo es paralela a la primera restricción del problema de Programación Lineal que se deseaba resolver. Puesto que se trataba de una desigualdad del tipo $\leq$, su valor máximo se alcanza precisamente para el valor de su recurso, 18.

6. En absoluto. Para saberlo es preciso fijarse en las curvas de isocoste puesto que son las que indican la dirección en las que están los extremos susceptibles de contener la solución óptima. En el caso de la maximización las funciones de coste tienden a desplazarse hacia las regiones con mayores valores de x_1 y x_2 (si tienen factores positivos en la función de coste) y al contrario en el caso de la minimización.

   Por lo tanto, cambiando de maximización a minimización en la función de coste se cambia completamente la parte de la región factible que debe examinarse y, en general, las soluciones son diferentes.

7. El problema dual resulta de *transponer* la matriz de coeficientes y de intercambiar los vectores de coste y recursos independientes. Además, la nueva función de coste será de minimización si la del problema primal era de maximización y al contrario. Por último, las restricciones serán ahora del tipo $\geq$:

$$\begin{array}{c} \text{mín}\, w = 18x'_1 + 8x'_2 \\ \begin{array}{cccccc} 2x'_1 & - & x'_2 & - & 5x'_3 & \geq & 2 \\ x'_1 & + & x'_2 & - & 2x'_3 & \geq & 1 \end{array} \\ x'_1, x'_2, x'_3 \geq 0 \end{array}$$

   de modo que ahora hay tantas variables como restricciones había antes y tantas restricciones como variables había en el problema primal.

8. En vez de aplicar el SIMPLEX a la resolución del problema de Programación Lineal mostrado en el apartado anterior es más razonable calcular la solución directamente a partir de la solución del problema primal ya calculada previamente.

   En particular, para resolver el problema de esta manera, se usará un teorema que afirma que:

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

   En particular:

$$x'^* = (2 \quad 1 \quad 0) \begin{pmatrix} \frac{1}{3} & -\frac{1}{3} & 0 \\ \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{7}{3} & -\frac{1}{3} & -1 \end{pmatrix} = (1 \quad 0 \quad 0)$$

   si se usan el vector de costes y la base obtenida en el último paso. Obviamente, tomando la base del paso anterior se habría obtenido la misma solución:

$$x'^* = (2 \quad 0 \quad 0) \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{5}{2} & 0 & -1 \end{pmatrix} = (1 \quad 0 \quad 0)$$

   En otras palabras, si el problema primal tiene soluciones infinitas, también es así en el problema dual. En particular, éstas se disponen entre los puntos duales de los puntos extremos primales como se acaba de mostrar.

9. Para la resolución del último 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

de modo que el valor óptimo de la primera variable, $x_1'^*$ es el valor pedido, 1.

Solución al problema 2

1. Un literal es *puro* cuando no aparece nunca negado en la fórmula F. Éste es el caso de los literales $\overline{x}_2$ y x_6. El primero aparece en las cláusulas C_1, C_3 y C_6. El segundo aparece únicamente en la última cláusula.

   Obviamente, una fórmula F en Forma Normal Conjuntiva será satisfacible y sólo sí lo es la fórmula que resulta de eliminar todos los literales puros. Por lo tanto, la resolución de la fórmula F respecto de los literales $\overline{x}_2$ y x_6 consiste simplemente en suprimir las cláusulas en las que aparecen, de modo que ahora consiste en la conjunción de las cláusulas:

   $$\begin{array}{ll} & C_4:\ (\overline{x}_3 \vee \overline{x}_4 \vee x_5) \\ C_2:\ (\overline{x}_1 \vee x_3) & C_5:\ (x_4 \vee \overline{x}_5) \end{array}$$

   En lo sucesivo se mantendrá la misma nomenclatura para nombrar las cláusulas, habida cuenta que en ningún apartado se hará referencia ya a las cláusulas del enunciado a menos que se indique lo contrario.

2. El algoritmo de Davis-Putnam se aplica en dos pasos. En el primero se aplica la resolución de variables escogidas iterativamente sobre el conjunto de cláusulas del paso anterior, hasta que por fin se alcanza el conjunto vacío, $\varnothing$ (en cuyo caso la fórmula original F es satisfacible) o la cláusula vacía, $\{\varnothing\}$, en cuyo caso la fórmula original F es no satisfacible.

   Si la fórmula original resulta ser satisfacible, entonces se asignan valores $\top$ (cierto) o $\bot$ (falso) a las variables escogidas en la primera fase, en el orden inverso en que fueron aplicadas, considerando únicamente las cláusulas a las que aplicaban entonces.

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

   Tal y como advertía el enunciado, se eligen las variables de la fórmula F (que son x_1, x_3, x_4 y x_5) en el orden ascendente de su subíndice. De modo que inicialmente se elige la variable x_1 y se almacena la selección en un vector dedicado `varSelect`:

   $\text{varSelect}[0] = x_1$ — *varSelect almacena la selección de variables por paso, y layerSeq las cláusulas*

   $$\begin{aligned} \text{layerSeq}[0] &= G_0 \backslash \text{Res}(G_0, x_1) = \\ &= G_0 \backslash \{C_4, C_5\} = \\ &= \{C_2\} \end{aligned}$$

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

   A continuación se considera la siguiente variable en el orden indicado, x_3

   $\text{varSelect}[1] = x_3$

   $$\begin{aligned} \text{layerSeq}[1] &= G_1 \backslash \text{Res}(G_1, x_3) = \\ &= G_1 \backslash \{C_5\} = \\ &= \{C_4\} \end{aligned}$$

   Paso 2 $G_2 = \{C_5\}$

   La siguiente variable a considerar es x_4

   $\text{varSelect}[2] = x_4$

   $$\begin{aligned} \text{layerSeq}[2] &= G_2 \backslash \text{Res}(G_2, x_4) = \\ &= G_2 \backslash \varnothing = \\ &= \{C_5\} \end{aligned}$$

Iteración	2	1	0
varSelect^{-1}	x_4	x_3	x_1
layerSeq^{-1}	$\{C_5\}$	$\{C_4\}$	$\{C_2\}$
Asignación	$x_4 = \top$	$x_3 = \bot$	$x_1 = \bot$

Cuadro 1: Asignación de valor a variables en el algoritmo Davis-Putnam

Como se obtuvo el conjunto vacío después de la última resolución, la fórmula F es satisfacible. Se procede ahora a asignar valores a las variables de la misma fórmula en orden inverso en que fueron consideradas.

La Tabla 1 muestra la asignación de valores (cierto, $\top$ o falso, $\bot$) a cada una de las variables de la fórmula F. En el primer paso (iteración 2), sólo debe considerarse la variable x_4 en la cláusula $C_5 : (x_4 \vee \overline{x_5})$. Obviamente, sólo puede satisfacerse con la asignación $x_4 = \top$ como se muestra en la tabla. En la iteración 1, debe elegirse una asignación de valor para la variable x_3, considerando únicamente la cláusula $C_4 : (\overline{x}_3 \vee \overline{x}_4 \vee x_5)$. Como $x_4 = \top$, la única manera de satisfacerla es haciendo $x_3 = \bot$. Por último, en la iteración 0 debe completarse el modelo que satisfará toda la fórmula eligiendo un valor para x_1 en la cláusula $C_2 : (\overline{x}_1 \vee x_3)$. Como en la iteración 1 se hizo $x_3 = \bot$, la única forma de satisfacer esta cláusula es con $x_1 = \bot$.

En conclusión, el modelo encontrado por el algoritmo de Davis-Putnam que satisface la fórmula original es:

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

Nótese, sin embargo, que la variable x_5 que aparecía en la fórmula que resulta de hacer la conjunción de las cláusulas consideradas en este apartado no tiene ningún valor asignado. En particular, puede tomar cualquiera de los valores $\bot$ o $\top$.

Asimismo, merece la pena mencionar que la solución del primer apartado resulta de extender esta solución con las únicas asignaciones posibles después de haber eliminado los literales puros, de modo que hubiera quedado:

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

3. Durante el segundo paso del algoritmo Davis-Putnam pudo observarse que, en todas las iteraciones, sólo había una asignación factible de valor a las variables. Ahora bien, el algoritmo de Davis-Putnam no dió valor a la variable x_5 que entonces, tal y como ya se dijo, puede tomar cualesquiera de los dos valores. Existen, por lo tanto, dos soluciones diferentes.

$$M_1 = \{x_1 = \bot, x_3 = \bot, x_4 = \top, x_5 = \bot\}$$
$$M_2 = \{x_1 = \bot, x_3 = \bot, x_4 = \top, x_5 = \top\}$$

Más aún, puesto que las variables que se eliminaron en el apartado 1 eran puras, también puede decidirse la asignación de valor a aquellas variables. En particular, para satisfacer la fórmula original del enunciado es obligatorio extender el modelo del apartado anterior con $\{x_2 = \bot, x_6 = \top\}$ y, otra vez, existen dos modelos diferentes, distinguidos por el valor de la variable x_5:

$$M_1 = \{x_1 = \bot, x_2 = \bot, x_3 = \bot, x_4 = \top, x_5 = \bot, x_6 = \top\}$$
$$M_2 = \{x_1 = \bot, x_2 = \bot, x_3 = \bot, x_4 = \top, x_5 = \top, x_6 = \top\}$$

4. El algoritmo de Davis-Putnam-Logemann-Loveland genera un árbol que enumera todas las posibles asignaciones de verdadero ($\top$) y falso ($\bot$) a las variables involucradas en la fórmula que se examina. En particular, el árbol generado expande los nodos en el orden de el primero en profundidad puesto que tiene profundidad acotada y así se consigue una ocupación de memoria lineal.

   Por lo tanto, el caso óptimo será aquel en el que el modelo se encuentra en el nodo más a la izquierda del árbol generado por el algoritmo DPLL. La ordenación pedida es, exactamente, el modelo encontrado en el apartado 2.

   La figura 1 muestra el árbol resultante. Nótese que en cada paso se reduce la fórmula del nodo considerado con la evaluación parcial indicada en cada arco: si una cláusula contiene el literal afirmado en el sentido indicado por la evaluación parcial entonces la cláusula desaparece; en otro caso, desaparece el literal y se

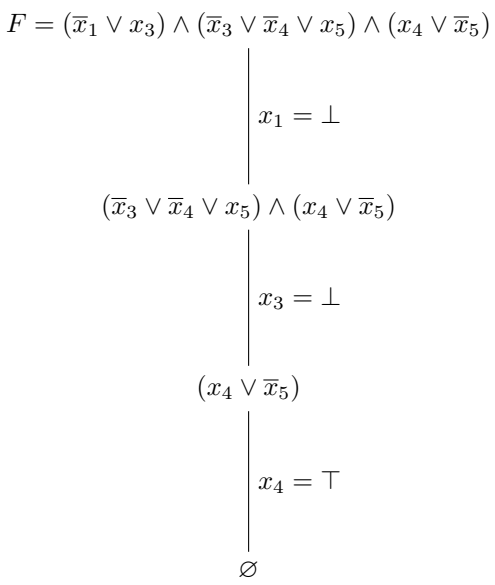


Figura 1: Algoritmo DPLL

preservan el resto de literales de la misma cláusula. Este proceso se conoce como *resolución unitaria* donde cada nueva fórmula resulta de hacer la resolución de cada cláusula de la fórmula precedente con la cláusula (de tamaño 1) en que consiste la asignación indicada en cada arco.

Como en el caso del algoritmo de Davis-Putnam, el único nodo hoja es el conjunto vacío con lo que se constata que la evaluación propuesta satisface la fórmula original.

Solución al problema 3

Tal y como se explica en el enunciado, este problema es uno de aquellos en los que debe hacerse un recorrido conectando nodos de un grafo consecutivamente hasta que por fin se han recorrido todos. Se trata, en esencia, de un problema muy parecido al TSP (*Travelling Salesperson Problem*). Sin embargo, no debe confundirse con el de generación de caminos *hamiltonianos*, puesto que no se pide que el recorrido acabe en el primer punto.

Aunque no se pedía explícitamente en el enunciado, una excelente práctica consiste en modelizar el espacio de estados. A continuación se discute primero la representación de los estados y luego la de los operadores antes de iniciar la resolución de los apartados del enunciado.

El problema consiste en modelizar el comportamiento de un **brazo**, cuyo trabajo consiste en concatenar **puntos** distribuidos sobre una **placa**.

Obviamente, los puntos estarán caracterizados por sus coordenadas (x, y). Además, los puntos deben estar convenientemente inscritos en el rectángulo de $M \times N$ que es la placa y cuyas dimensiones se representan simplemente con el par (N, M).

Como el brazo recorrerá los puntos de la placa con el propósito de unirlos, su posición también está caracterizada por un par ordenado (x, y). Por lo tanto, la posición del robot se representará con un par (r_x, r_y).

Aunque el enunciado del problema describe varias acciones (como hacer descender el brazo del robot hasta la placa, o levantarlo desde ella al finalizar el trabajo), lo cierto es que sólo hay una acción relevante para el enunciado del problema, y ésta es mover el brazo desde una posición hasta otra.

En este punto debe decidirse ahora cómo mover el brazo. ¿Es mejor decidir el movimiento desde una posición hasta otra adyacente en la placa? o por el contrario, ¿será mejor ejecutar el movimiento del brazo desde un punto hasta otro (recorriendo así una cantidad arbitraria de puntos intermedios)? Mientras que la primera alternativa resulta en un operador con coste siempre igual (puesto que sólo hace desplazamientos unitarios, de longitud uno), la segunda alternativa resulta en una implementación más eficiente donde el coste del operador es igual a la longitud del camino que describe. El motivo por el que el coste es igual a la longitud de ese camino es porque en el enunciado se advierte explícitamente que se desea ahorrar en el material empleado para concatenar todos los puntos. La distancia entre dos puntos $P_0(x_0, y_0)$ y $P_1(x_1, y_1)$ se expresa como la distancia de Manhattan entre ellos tal y como se indica a continuación:

`Mover` $((r_x, r_y), (x, y))$:

SI $(x < M) \wedge (y < N) \wedge$	*Si el punto destino*
$(x \geq 0) \wedge (y \geq 0)$	*está inscrito en la placa*
ENTONCES $r_x = x \wedge r_y = y$	*actualiza la posición de la cabeza*
$k = \lvert r_x - x \rvert + \lvert r_y - y \rvert$	*(coste del operador)*

donde k representa el coste de aplicar el operador `Mover` que desplaza la cabeza desde su posición actual (r_x, r_y) hasta la posición destino (x, y).

A continuación, se discute la solución de los apartados del problema.

1. Puesto que el operador definido en el apartado anterior tiene un coste no unitario, la alternativa más razonable consiste en elegir uno entre los algoritmos de búsqueda de fuerza bruta con costes. En concreto, podría elegirse entre el algoritmo de *Dijkstra* (que es un algoritmo de el mejor primero con $f(n) = g(n)$), o el algoritmo de *ramificación y acotación en profundidad*. Cualquiera de ellos garantiza que encontrará una solución óptima (esto es, son *admisibles*). Además, mientras que el primer algoritmo tiene una ocupación de memoria exponencial (pero no reexpande nodos), el segundo tiene una ocupación de memoria lineal (aunque reexpande nodos).

   Por lo tanto, el algoritmo de Dijkstra puede ser más adecuado para aquellos problemas sencillos donde la enumeración de todos los estados posibles cabe en memoria. Sin embargo, el segundo es capaz de resolver los problemas más difíciles.

2. Puesto que el coste del operador definido en el segundo apartado es igual a la distancia de Manhattan, lo más razonable será *estimar* ahora el coste con esa misma expresión. Puesto que la distancia de Manhattan es normalmente mayor que la distancia euclídea, no tendría ningún sentido usar esta segunda —puesto que resultaría en una estimación menos informada.

   Ahora bien, dado un estado n en el que ya se han unido una cantidad arbitraria de nodos u, de modo que faltan hasta v por conectar[2], ¿cuál es la mejor estimación del esfuerzo restante?

   Una alternativa (demasiado conservadora) sería tomar la distancia de Manhattan al punto más cercano a la posición actual del brazo:

   $$h_1 = \min_{i=1,v} \{\lvert r_x - x_i \rvert + \lvert r_y - y_i \rvert\}$$

   Pero, en realidad, ¡deben conectarse todos los puntos!, de modo que la distancia de Manhattan hasta el punto más alejado, no sólo será admisible sino que, además, será un valor mejor informado que el de h_1:

   $$h_2 = \max_{i=1,v} \{\lvert r_x - x_i \rvert + \lvert r_y - y_i \rvert\}$$

   En ambos casos, (x_i, y_i) representa el par de coordenadas del punto i-ésimo.

   Por supuesto, aún es posible computar otras distancias admisibles y mejor informadas. Por ejemplo, bastaría con calcular los saltos a los **dos** puntos más alejados, en vez del más alejado, puesto que, como antes, ¡deben recorrerse todos! Sin embargo, este cálculo crece exponencialmente puesto que deben considerarse todas las combinaciones posibles.

3. En este caso, la distancia de Manhattan podría sobreestimar fácilmente la distancia entre dos puntos. Por lo tanto, debe usarse la distancia euclídea entre dos puntos $P_r(r_x, r_y)$ y $P(x, y)$ definida como sigue:

   $$d_E = \sqrt{(r_x - x)^2 + (r_y - y)^2}$$

4. La respuesta es que no, y el motivo es que si hay una cabeza distribuyendo material para conectar dos puntos, ese mismo material podría ser utilizado por la otra cabeza tal y como se advertía en el enunciado. Por lo tanto, la cantidad de material bien podría ser **menor** que la distancia de Manhattan o la distancia euclídea. En otras palabras, las distancias heurísticas de los apartados anteriores bien podrían sobreestimar la cantidad de material a emplear y, por lo tanto, resultar en estimaciones *no admisibles*.

[2] De modo que, obviamente, $u + v = N$