Departamento de Informática
Universidad Carlos III de Madrid

Grado en Informática

Heurística y Optimización

Enero 2013

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. (4,5 puntos)

Considera el siguiente problema de Programación Lineal:

$$\begin{array}{rcrcrcl} & & \max z = x_1 + 2x_2 - 3x_3 & & & & \\ 2x_1 & + & x_2 & - & x_3 & \leq & 4 \\ x_1 & + & x_2 & + & 2x_3 & \geq & 3 \\ & & \mathbf{x} \geq \mathbf{0} & & & & \end{array}$$

y responde razonadamente las siguientes preguntas:

1. **(0,5 puntos)** Expresa el problema de Programación Lineal anterior en forma *estándar* de maximización.
2. **(0,5 puntos)** Prepara el problema de Programación Lineal que ha resultado del apartado anterior para su aplicación con el algoritmo Simplex para garantizar que pueda comenzarse su aplicación con la matriz identidad.
3. **(1 punto)** Resuelve el problema de Programación Lineal obtenido en el apartado anterior con el algoritmo Simplex. Indica claramente, en cada iteración: las variables escogidas en la base, su valor, y el valor de la función objetivo.
4. **(0,5 puntos)** Interpreta el resultado y explica qué conclusiones pueden extraerse de él.
5. **(0,5 puntos)** ¿Cuál es el problema dual del problema de Programación Lineal original?
6. **(1 punto)** Resuelve el problema dual indicando claramente el valor de todas las variables de decisión duales y el valor óptimo de la función objetivo dual.
7. **(0,5 puntos)** ¿Cuál es la contribución por unidad del recurso que se corresponde con la segunda restricción al valor óptimo de la función de coste?

Problema 2. (2 puntos)

Considera las siguientes reacciones químicas:

$$MgO + H_2 \longrightarrow Mg + H_2O$$
$$C + O_2 \longrightarrow CO_2$$
$$CO_2 + H_2O \longrightarrow H_2CO_3$$

considerando ahora que hay siempre una cantidad suficiente de MgO, H_2, O_2 y C responde razonadamente las siguientes preguntas:

1. **(0,5 puntos)** Modeliza las reacciones químicas mostradas con una fórmula lógica F en Forma Normal Conjuntiva
2. **(0,5 puntos)** Resuelve el problema resultante con el algoritmo de Davis-Putnam-Logemann-Loveland

Considera ahora la siguiente fórmula lógica:

$$F \equiv (p \vee r \vee \overline{s}) \wedge (\overline{p} \vee \overline{s}) \wedge (\overline{q} \vee \overline{r}) \wedge (q \vee s)$$

Se pide:

3. **(1 punto)** Encuentra un modelo, si existe, para la fórmula F usando el algoritmo de Davis-Putnam.

Problema 3. (3,5 puntos)

Un instituto científico desea dotar a su sistema multiprocesador de un software inteligente que sirva para distribuir varios trabajos de cómputo entre sus procesadores.

Los trabajos que deben hacerse son conocidos *a priori*, y de cada uno de ellos se sabe con precisión el tiempo que requieren para llevarse a cabo en un solo procesador, sin interrupciones, desde el principio hasta su conclusión. Asimismo, el sistema multiprocesador está dotado de un número finito no ampliable de procesadores, cada uno de los cuales puede ser usado durante un intervalo de tiempo con la condición de que no esté ocupado con el procesamiento de otro trabajo. Cada procesador está indefinidamente disponible para el procesamiento de cualquier trabajo.

Se desea diseñar un sistema inteligente que sirva para minimizar el *makespan* o tiempo completo de procesamiento de los trabajos cuando se considera la asignación en paralelo a varios procesadores.

Se pide:

1. **(0,5 puntos)** Formalizar el espacio de estados
2. **(0,5 puntos)** ¿Qué algoritmo de búsqueda de fuerza bruta usarías para resolver este problema si se desean soluciones óptimas? Razona tu respuesta.
3. **(1 punto)** Obtener una función heurística *admisible* e *informada* que sirva para minimizar el *makespan*.
4. **(0,5 puntos)** ¿Qué algoritmo de búsqueda sugerirías emplear? Razona tu respuesta.
5. Recientemente, el instituto científico ha recibido un número determinado de procesadores que van todos hasta p veces más rápido que los antiguos. Considerando su uso concurrente con los procesadores anteriores, se pide:
   - *a*) **(0,5 puntos)** ¿Es preciso hacer alguna modificación en la representación del espacio de problemas sugerida en el apartado 1? Si es así, ¿cuál o cuáles?
   - *b*) **(0,5 puntos)** ¿Es preciso hacer alguna modificación en el cálculo de la función heurística del apartado 3? Si es así, ¿cuál o cuáles?

Soluciones del examen de Heurística y Optimización
Enero 2013

Solución al problema 1

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

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

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

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

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

   donde se han añadido las variables de holgura x_4 y x_5 de acuerdo con las observaciones anteriores. Además, las variables de holgura se añaden a la función objetivo con coeficiente nulo de modo que pueden ignorarse.

2. Para poder empezar a aplicar el algoritmo SIMPLEX es preciso disponer de una base factible inicial que, por lo tanto, debe verificar:

   a) La base elegida B debe ser invertible. Esto es $|B| \neq 0$

   b) El producto de la inversa de la base, B^{-1}, por el vector de recursos $\mathbf{b}$ debe dar valores no negativos

   Normalmente, una buena elección para iniciar la aplicación del algoritmo SIMPLEX consiste en elegir la matriz identidad (de la dimensión adecuada) puesto que: primero, es evidentemente invertible; segundo, su inversa, siendo igual a sí misma y multiplicada por el vector de recursos $\mathbf{b}$ sólo puede dar términos no negativos si el problema está en forma estándar porque, como se explicó en el primer apartado, el vector de recursos debe consistir únicamente de términos no negativos.

   Ahora bien, no existe ninguna selección de vectores columna a_i en la matriz de coeficientes del problema de Programación Lineal obtenido en el apartado anterior que sea igual a la matriz identidad de dimension 2, I_2. Sin embargo, escogiendo a_4 ya se tiene la primera columna de I_2. Añadiendo ahora una *variable artificial* x_6 a la segunda restricción ya se tiene por completo la matriz identidad I_2 formada entonces por a_4 y a_6. Como las variables artificiales son sólo de carácter técnico, deben añadirse a la función objetivo con el coeficiente $-\infty$ —y, en caso de que la solución del SIMPLEX asigne valores no negativos a alguna variable artificial, entonces el problema es irresoluble.

   Por lo tanto, el problema de Programación Lineal queda ahora de la forma:

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

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

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

Paso 0 Cálculo de una solución factible inicial

a) Cálculo de las variables básicas
La primera iteración se inicia con una base igual a la matriz identidad de dimensión 2, tal y como se pedía en el enunciado. Por lo tanto, son variables básicas en este paso $\{x_4, x_6\}$

$$B_0 = I_2 \qquad B_0^{-1} = I_2$$
$$x_0^* = B_0^{-1}b = b = \begin{pmatrix} 4 \\ 3 \end{pmatrix} \quad z_0^* = c_{B_0}^T x_0^* = \begin{pmatrix} 0 & -\infty \end{pmatrix} \begin{pmatrix} 4 \\ 3 \end{pmatrix} = -\infty$$

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

$$z_1 - c_1 = c_{B_0}^T B_0^{-1} a_1 - c_1 = \begin{pmatrix} 0 & -\infty \end{pmatrix} I_2 \begin{pmatrix} 2 \\ 1 \end{pmatrix} - 1 = -\infty - 1$$
$$z_2 - c_2 = c_{B_0}^T B_0^{-1} a_2 - c_2 = \begin{pmatrix} 0 & -\infty \end{pmatrix} I_2 \begin{pmatrix} 1 \\ 1 \end{pmatrix} - 2 = -\infty - 2$$
$$z_3 - c_3 = c_{B_0}^T B_0^{-1} a_3 - c_3 = \begin{pmatrix} 0 & -\infty \end{pmatrix} I_2 \begin{pmatrix} -1 \\ 2 \end{pmatrix} + 3 = -2\infty + 3$$
$$z_5 - c_5 = c_{B_0}^T B_0^{-1} a_5 - c_5 = \begin{pmatrix} 0 & -\infty \end{pmatrix} I_2 \begin{pmatrix} 0 \\ -1 \end{pmatrix} - 0 = +\infty$$

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

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

y sale la variable x_6 como cabía esperar puesto que al ser una variable artificial tiene una penalización arbitrariamente grande en la función de coste.

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

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

$$B_1 = \begin{pmatrix} -1 & 1 \\ 2 & 0 \end{pmatrix} \qquad B_1^{-1} = \begin{pmatrix} 0 & \frac{1}{2} \\ 1 & \frac{1}{2} \end{pmatrix}$$
$$x_1^* = B_1^{-1}b = \begin{pmatrix} \frac{3}{2} \\ \frac{11}{2} \end{pmatrix} \quad z_1^* = c_{B_1}^T x_1^* = \begin{pmatrix} -3 & 0 \end{pmatrix} \begin{pmatrix} \frac{3}{2} \\ \frac{11}{2} \end{pmatrix} = -\frac{9}{2}$$

b) Selección de la variable de entrada

$$z_1 - c_1 = c_{B_1}^T B_1^{-1} a_1 - c_1 = \begin{pmatrix} -3 & 0 \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{2} \\ 1 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} - 1 = -\frac{5}{2}$$
$$z_2 - c_2 = c_{B_1}^T B_1^{-1} a_2 - c_2 = \begin{pmatrix} -3 & 0 \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{2} \\ 1 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} - 2 = -\frac{7}{2}$$
$$z_5 - c_5 = c_{B_1}^T B_1^{-1} a_5 - c_5 = \begin{pmatrix} -3 & 0 \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{2} \\ 1 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 0 \\ -1 \end{pmatrix} - 0 = \frac{3}{2}$$

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

$$\text{mín}\left\{\frac{\frac{3}{2}}{\frac{1}{2}}, \frac{\frac{11}{2}}{\frac{3}{2}}\right\}$$

de modo que abandona la base la variable x_3

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

a) Cálculo de las variables básicas
De nuevo, se vuelve a intentar mejorar la solución del paso anterior con las variables básicas $\{x_2, x_4\}$

$$B_2 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \qquad B_2^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix}$$
$$x_2^* = B_2^{-1}b = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \quad z_2^* = c_{B_2}^T x_2^* = \begin{pmatrix} 2 & 0 \end{pmatrix}\begin{pmatrix} 3 \\ 1 \end{pmatrix} = 6$$

b) Selección de la variable de entrada

$$z_1 - c_1 = c_{B_2}^T B_2^{-1} a_1 - c_1 = \begin{pmatrix} 2 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 2 \\ 1 \end{pmatrix} - 1 \;\; = 1$$
$$z_3 - c_3 = c_{B_2}^T B_2^{-1} a_3 - c_3 = \begin{pmatrix} 2 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} + 3 \;\; = 7$$
$$z_5 - c_5 = c_{B_2}^T B_2^{-1} a_5 - c_5 = \begin{pmatrix} 2 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ -1 \end{pmatrix} - 0 = -2$$

Nótese que no se ha calculado el coste reducido de x_6 puesto que al restar su coeficiente en la función de coste, resultaría $+\infty$ y, por lo tanto, no podría ser negativo.

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

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

de donde resulta que la variable reemplazada será x_4

Paso 3 Mejora de la solución actual (iteración #3)

a) Cálculo de las variables básicas
En esta iteración, las variables básicas son $\{x_2, x_5\}$

$$B_3 = \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix} \qquad B_3^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}$$
$$x_3^* = B_3^{-1}b = \begin{pmatrix} 4 \\ 1 \end{pmatrix} \quad z_3^* = c_{B_3}^T x_3^* = \begin{pmatrix} 2 & 0 \end{pmatrix}\begin{pmatrix} 4 \\ 1 \end{pmatrix} = 8$$

b) Selección de la variable de entrada

$$z_1 - c_1 = c_{B_3}^T B_3^{-1} a_1 - c_1 = \begin{pmatrix} 2 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 2 \\ 1 \end{pmatrix} - 1 \;\; = 3$$
$$z_3 - c_3 = c_{B_3}^T B_3^{-1} a_3 - c_3 = \begin{pmatrix} 2 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} -1 \\ 2 \end{pmatrix} + 3 = 1$$
$$z_4 - c_4 = c_{B_3}^T B_3^{-1} a_4 - c_4 = \begin{pmatrix} 2 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} - 0 \;\; = 2$$

Otra vez, no se ha calculado el coste reducido de x_6 puesto que al restar su coeficiente en la función de coste, resultaría $+\infty$ y, por lo tanto, no podría ser negativo.
Como todos los costes reducidos tienen valores positivos, se acaba con la solución óptima:

$$x^* = \begin{pmatrix} 0 \\ 4 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad z^* = 8$$

4. En primer lugar, la solución es factible porque no ha otorgado ningún valor positivo a la variable artificial x_6. La solución óptima obtenida, entonces, debe leerse como una inversión de hasta 4 unidades en el concepto representado en x_2 de donde resulta un sobrante en la segunda restricción de 1 unidad de acuerdo con el valor de la variable de holgura x_5. Por último, es importante enfatizar que la solución es única puesto que los *costes reducidos* obtenidos en el último paso son todos estrictamente positivos —y esto se entiende como que cualquier variación en la base decrementaría el valor de la función objetivo.

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

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

de donde resulta que el problema dual es:

$$\begin{array}{c} \text{mín } w = 4x'_1 - 3x'_2 \\ \begin{array}{rcrcrcr} 2x'_1 & - & x'_2 & \geq & 1 \\ x'_1 & - & x'_2 & \geq & 2 \\ - \quad x'_1 & - & 2x'_2 & \geq & -\ 3 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

6. Por supuesto, en este apartado es posible iniciar la aplicación de otro SIMPLEX para resolver el problema de Programación Lineal del 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.

En particular:

$$x'^* = c_B^T B^{-1} = (2 \quad 0) \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix} = (2 \quad 0)$$

A partir de esta solución, el valor óptimo del problema dual se calcula como un producto matricial como de costumbre:

$$w^* = c_w x'^* = \begin{pmatrix} 4 & -3 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \end{pmatrix} = 8$$

7. 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 segunda variable, $x_2'^*$ es el valor pedido, 0.

Solución al problema 2

1. Como en todos los casos de modelización de lógica proposicional cada concepto se representa con una proposición. Para ello, se sugieren las siguientes:

$$\begin{array}{llll} x_1{\sim}MgO & x_2{\sim}H_2 & x_3{\sim}\,Mg & x_4{\sim}\;\; H_2O \\ x_5{\sim}\;\; C & x_6{\sim}O_2 & x_7{\sim}CO_2 & x_8{\sim}H_2CO_3 \end{array}$$

donde cada x_i representa la disponibilidad de los elementos indicados en cada caso.

A continuación se discuten dos alternativas diferentes para la modelización del primer problema:

Primera alternativa La primera posibilidad consiste en implementar las reacciones químicas como si se tratase de implicaciones que deben entenderse como que la existencia simultánea de los elementos indicados a la izquierda necesariamente produce aquellos indicados a la derecha.
Por ejemplo, la primera reacción se modelizaría entonces como sigue:

$$(x_1 \wedge x_2) \longrightarrow (x_3 \wedge x_4)$$

que, puede reescribirse por definición de la implicación lógica:

$$\overline{(x_1 \wedge x_2)} \vee (x_3 \wedge x_4)$$

y aplicando las reglas de Morgan:

$$(\overline{x_1} \vee \overline{x_2}) \vee (x_3 \wedge x_4)$$

Por último, aplicando ahora la regla distributiva resulta:

$$(\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$$

que ya está conveniente representada en Forma Normal Conjuntiva.
Las otras dos reglas son más sencillas y se obtienen por aplicación directa de la definición de implicación lógica y son:

$$\begin{array}{c} (\overline{x_5} \vee \overline{x_6} \vee x_7) \\ (\overline{x_7} \vee \overline{x_4} \vee x_8) \end{array}$$

Por lo tanto, resultaría una fórmula F_1 en Forma Normal Conjuntiva que consiste en la conjunción de las siguientes cláusulas:

$$\begin{array}{l} C_1{:}(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ C_2{:}(\overline{x_1} \vee \overline{x_2} \vee x_4) \\ C_3{:}(\overline{x_5} \vee \overline{x_6} \vee x_7) \\ C_4{:}(\overline{x_7} \vee \overline{x_4} \vee x_8) \end{array}$$

Nótese, sin embargo, que esta modelización se refiere a las producciones de elementos químicos a partir de las interacciones con otros. En particular, si los reactivos existen, la fórmula se satisface si y sólo si se generan los productos como debe ser. Sin embargo, esta fórmula admite también modelos donde los reactivos no están disponibles. En particular, si ningún reactivo de las dos primeras reacciones químicas está disponible, tal y como se especifica en el siguiente modelo:

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

la fórmula se valida igualmente. Sin embargo, el enunciado advertía explícitamente que todos los componentes están disponibles

Segunda alternativa En la segunda alternativa consiste en intentar, explícitamente que la fórmula resultante tenga modelos si y sólo si se verifican en el problema original. En particular, no se desean modelos que representen la no disponibilidad de elementos puesto que este caso no puede darse.

Considerando ahora entonces que hay cantidades suficientes de MgO, H_2, O_2 y C, es obvio que H_2CO_3 se produce. Esto es: $F_2 \equiv x_8$ es una fórmula que modeliza el problema aunque no tiene en cuenta los reactivos para su consecución.

Ahora bien, si x_8 es cierto, entonces lo serán los reactivos que lo producen y la misma fórmula puede reescribirse como $F_2 \equiv x_4 \wedge x_7$.

Análogamente, lo mismo puede decirse de las proposiciones x_4 y x_7 que se sustituyen, conveniente por la conjunción de las moléculas que se necesitan para su producción resultando entonces la fórmula:

$$F_2 \equiv x_1 \wedge x_2 \wedge x_5 \wedge x_6$$

que ya está convenientemente escrita en Forma Normal Conjuntiva.

Nótese que esta expresión sólo puede validarse si y sólo si existen todos los reactivos necesarios tal y como se especificaba en el enunciado y no lo hace si no están disponibles.

2. A continuación se resuelven las dos fórmulas obtenidas anteriores F_1 y F_2 (en los puntos *Primera alternativa* y *Segunda alternativa* respectivamente) resolviendo así este apartado también de dos formas diferentes. En cualquier caso, primero se eliminan las cláusulas en las que haya literales puros y sólo si resulta una fórmula no vacía se aplica el algoritmo de Davis-Putnam.

   Primera alternativa Los literales, $\overline{x_1}$, $\overline{x_2}$ y x_3 son puros. Por lo tanto, pueden eliminarse las dos primeras cláusulas después de establecer valores para x_1, x_2 y x_3 acordes con el signo con el que aparecen: $\{x_1 = \bot, x_2 = \bot, x_3 = \top\}$.

   Observando ahora las cláusulas resultantes (C_3 y C_4) se advierte que $\overline{x_4}$, $\overline{x_5}$, $\overline{x_6}$ y x_8 son también literales puros por lo que pueden eliminarse después de extender el modelo anterior con las siguientes asignaciones $\{x_4 = \bot, x_5 = \bot, x_6 = \bot, x_8 = \top\}$.

   Después de eliminar todas las cláusulas resulta una fórmula vacía que es satisfacible y no hay necesidad de aplicar Davis-Putnam.

   Segunda alternativa Resulta obvio que todos los literales en F_2 son puros de modo que al eliminar las cláusulas en las que aparecen resulta de inmediato la fórmula vacía, de modo que no hay necesidad de aplicar el algoritmo de Davis-Putnam y el único modelo que existe es:

$$M_2 = \{x_1 = \top, x_2 = \top, x_5 = \top, x_6 = \top\}$$

3. Como en el apartado anterior, el primer paso consiste en observar si hay o no literales puros. Como no es el caso, las cláusulas que deben considerarse para la aplicación del algoritmo DPLL son las siguientes:

$$\begin{array}{ll} C_1{:} & (p \vee r \vee \overline{s}) \\ C_2{:} & (\overline{p} \vee \overline{s}) \\ C_3{:} & (\overline{q} \vee \overline{r}) \\ C_4{:} & (q \vee s) \end{array}$$

   A continuación se detallan los pasos del algoritmo DPLL que consiste en dos fases: en la primera se aplica repetidamente la resolución seleccionando una variable en cada paso hasta que se obtenga el vació o la cláusula vacía; si resultó el vacío, se procede entonces a crear un modelo en la segunda fase considerando las variables y cláusulas empleadas en cada paso de la primera fase en orden inverso.

   Paso 0 $G_0 = \{C_1, C_2, C_3, C_4\}$

   $\text{varSelect}[0] = p$ — *varSelect almacena la selección de variables por paso, y layerSeq las cláusulas nótese que se genera C_5*

   $\text{layerSeq}[0] = G_0 \backslash \text{Res}(G_0, p) = G_0 \backslash \{C_3, C_4, C_5 : (r \vee \overline{s})\} = \{C_1, C_2\}$

   Paso 1 $G_1 = \{C_3, C_4, C_5\}$

Paso	2	1	0
varSelect	r	q	p
layerSeq	$\{C_5, C_6\}$	$\{C_3, C_4\}$	$\{C_1, C_2\}$
Modelo 1	$\{r = \top, s = \top\}$	$q = \bot$	$p = \bot$
Modelo 2	$\{r = \bot, s = \bot\}$	$q = \top$	

Cuadro 1: Resultado de la segunda fase del algoritmo DPLL

$\text{varSelect}[1] = q$
$\text{layerSeq}[1] = G_1 \backslash \text{Res}(G_1, q) =$
$G_1 \backslash \{C_5, C_6 : (\overline{r} \vee s)\} =$ *ahora se ha generado* C_6
$\{C_3, C_4\}$

Paso 2 $G_2 = \{C_5, C_6\}$

$\text{varSelect}[2] = r$
$\text{layerSeq}[2] = G_2 \backslash \text{Res}(G_2, r) =$
$G_2 \backslash \varnothing =$ *como resulta una tautología no se añade nada*
$\{C_5, C_6\}$

de donde ha resultado el conjunto vacío. Por lo tanto, se procede a elaborar un modelo en la segunda fase del algoritmo DPLL cuyo resultado está resumido en la tabla 1. Nótese que la variable p puede tomar cualquier valor en el segundo modelo.

Solución al problema 3

1. La representación del espacio de problemas consiste en decidir, por una parte, la representación de los estados y, por la otra, de los operadores con los que es posible transitar desde unos estados hasta otros.

   Para la caracterización de los estados es importante elegir estructuras de datos que, convenientemente, representen los siguientes conceptos relevantes del enunciado:

   `Trabajos` Cada uno tendrá los siguientes atributos:

   - `id` Sirve para distinguir cada trabajo por un identificador único
   - `t` Indica la cantidad de tiempo necesaria para su ejecución
   - `asignado` Puede tomar los valores `si` y `no` y sirve para distinguir, en cada estado, los trabajos ya asignados de los que siguen pendientes

   `Procesadores` Caracterizados como sigue:

   - `id` Como en el caso de los trabajos, sirve para distinguir cada procesador de los demás
   - `t` Indica el instante de tiempo a partir del cual cada procesador está disponible. Como el enunciado advertía que están inmediatamente disponibles, su valor será 0 por defecto.

   Con estas descripciones, ya es posible crear otra estructura de datos que contendrá información del *schedule* que resuelve el problema:

   `Schedule` Contendrá la siguiente información relevante:

   - `makespan` , contiene la duración total del *schedule* generado en cada estado —y que se define como la diferencia de tiempo desde la finalización del último trabajo y el inicio del primer trabajo
   - `Trabajos` Consiste en una referencia a todas las instancias de `Trabajo` definidas más arriba. Análogamente, se añade a la definición de un `Schedule`:
   - `Procesadores` Una referencia a estructuras `Procesadores` por cada procesador disponible

   Efectivamente, la representación sugerida es suficiente para distinguir los estados iniciales y finales como se indica a continuación:

- En el estado inicial habrá tantas instancias de `Trabajo` como peticiones haya, distinguidos por su `id` y con el valor de `asignado` igual a `no`. Su atributo `t` tomará el valor que representa el tiempo que tardará en ejecutarse en un procesador.

  Por otra parte, habrá tantas instancias de `Procesador`, como procesadores haya en el sistema multiprocesador considerado, distinguidos todos por un `id`. Además, el atributo `t`, que representa la disponibilidad de un procesador, tomará el valor 0 en todas las instancias indicando, con ello, que están inmediatamente disponibles.

  Por otra parte, el `schedule` tendrá un `makespan` inicialmente igual a 0.
- En el estado final, todos los trabajos deberán tener el atributo `asignado` igual a `si`. Después del procesamiento, el slot `makespan` del `schedule` tendrá un valor distinto de 0 que es, de hecho, el parámetro que se desea minimizar.

Por otra parte, el único operador de este problema resulta fácilmente de observar que la única acción contemplada en este problema consiste en asignar procesadores a la realización de trabajos:

`Asignar`:

SI $\exists t \in$ `Trabajo` $\wedge$
$\exists m \in$ `Procesador` $\wedge$
$\exists s \in$ `Schedule` $\wedge$
`asignado` (t) = `no`
ENTONCES dar-valor (t.`asignado`, `si`)
dar-valor (m.`t`,t.`t` + m.`t`)
dar-valor (s.`makespan`, máx$\{s$.`makespan`, m.`t` + t.`t`$\}$)
$k = s$.`makespan`

Obsérvese que el operador actualiza el coste de la solución parcial construída con su aplicación al `makespan` o tiempo del procesador que está más tiempo ocupado como se indica en el atributo k. Esto tiene dos consecuencias importantes:

- Los operadores no tienen costes unitarios. En otras palabras, se prefiere la ejecución de unos operadores a otros —en particular, es fácil observar que se prefieren las asignaciones en paralelo a aquellas que ocurren secuencialmente.
- Además, los costes de cada operador no son *aditivos*. Esto es, el coste de una solución parcial será igual al makespan calculado en el parámetro k y no la suma de los makespans de todos los antecesores de un nodo.

Ambas observaciones tienen consecuencias importantes en la solución de los apartados 2 y 4.

2. Con la consideración de costes mostrada en el operador `asignar` del apartado anterior no existe una asociación única entre la profundidad de una solución y su coste. Por lo tanto, el algoritmo del primero en amplitud queda descartado. Otros algoritmos admisibles para el tratamiento con costes son:

   Primero en profundización iterativo Si se inicia la ejecución del algoritmo con un coste igual al del tiempo de procesamiento del trabajo más largo (puesto que, al fin y al cabo debe atenderse como cualquier otro) y se actualiza el umbral de cada iteración al mínimo *makespan* de todos los nodos hoja de la iteración anterior este algoritmo encontraría soluciones óptimas

   Ramificación y acotación en profundidad Otra alternativa inteligente consiste en encontrar una primera solución y después refinarla podando aquellos nodos no terminales (esto es, que no son solución) que ya hayan excedido el *makespan* de la mejor solución conocida hasta ahora.

   La selección de cualquiera de estos algoritmos es razonable puesto que aunque son algoritmos del primero en profundidad, el número máximo de niveles que tienen que expandir está acotado y es igual al número de trabajos que deben procesarse

3. Resulta fácil observar que en la aplicación de la técnica de *relajación de restricciones*, las restricciones factibles de ser relajadas y que, de hecho, afectan al coste (como puede observarse en la formulación de la regla `asignar`) son:

- La secuencialidad en la asignación de trabajos a procesadores: podría relajarse esta condición imaginando entonces que cualquier procesador podría atender cualesquiera trabajos ¡al mismo tiempo! —lo que es obviamente infactible, pero puede que sea conveniente a efectos de calcular una función heurística. La solución óptima de cualquier problema en este caso sería (como en el caso del cálculo del primer umbral para el algoritmo de profundización iterativa en el apartado anterior) el tiempo máximo de procesamiento de cualquiera de los trabajos pendientes de ser `asignado`s:

$$h_1 = \max_{t \in \texttt{Trabajo} \wedge t.\texttt{asignado}=\texttt{no}} \{t.\texttt{t}\}$$

- El hecho de que los trabajos deben ser atendidos en un único procesador: esto es, podría imaginarse que cualquier trabajo pendiente de ser `asignado` es descompuesto en varias partes, cada una de las cuales sería entonces ejecutada por un procesador diferente. Esta suposición, es la que inevitablemente reduce el `makespan` al máximo:

$$h_2 = \min_{m \in \texttt{Procesador}} \left\{ m.\texttt{t} + \frac{1}{M} \sum_{i=1}^{T'} t_i.\texttt{t} \right\}$$

  donde M es el número de procesadores disponibles y T' es el número de trabajos pendientes de ser `asignado`s —nótese que, en el apartado anterior se había definido T como el número de trabajos definidos inicialmente, de modo que $T' \leq T$.

  Obviamente, h_2 será admisible. Más aún, en algún caso podría ser incluso menor que el `makespan` que ya se haya consumido en una solución parcial construida durante el procedimiento de búsqueda, de modo que para mejorar la estimación anterior se sugiere la función heurística siguiente:

$$h_3 = \max \left\{ \min_{m \in \texttt{Procesador}} \left\{ m.\texttt{t} + \frac{1}{M} \sum_{i=1}^{T'} t_i.\texttt{t} \right\} ; \texttt{makespan} \right\}$$

Claramente, de todas las funciones heurísticas construidas metódicamente con el modelo de *relajación de restricciones*, h_3 es la más informada y, por lo tanto, la que sugiere en la resolución de este problema.

4. A la hora de decidir qué algoritmo de búsqueda debería emplearse para resolver un problema concreto resulta conveniente analizar cuidadosamente las siguientes cuestiones:

   - ¿Se dispone de una función heurística? Si es así, entonces será posible aplicar la clase de algoritmos de búsqueda denominados *heurísticos*; en otro caso, sólo deberían considerarse los algoritmos de búsqueda *de fuerza bruta*.
   - En cualquier caso, también es preciso decidir si se desea recorrer el espacio de estados en *profundidad* o en *amplitud* ocupando más o menos espacio en el ordenador al tiempo que se persiguen soluciones óptimas o no.

   En este problema, se dispone efectivamente de una función heurística, de modo que se sugiere restringir la atención a la clase de algoritmos de búsqueda heurística. De entre ellos, llaman especialmente la atención los algoritmos de el mejor primero puesto que el algoritmo A* (esto es, el algoritmo de el mejor primero que resulta de aplicar $f(n) = g(n) + h(n)$) es *completo* y, además, *admisible*. Lo mismo cabe decirse del algoritmo IDA*. Sin embargo, para poder aplicar la función $f(n)$ descrita, es preciso que $g(n)$ y $h(n)$ puedan sumarse convenientemente.

   Obsérvese que, en este caso, $g(n)$ es la suma de los costes del operador `asignar` que, tal y como se ha definido, será la suma de varios `makespan` seguidos, ¡en vez del cálculo incremental de un único *makespan*!, mientras que $h(n)$ es la estimación de un único *makespan* para la realización de todos los trabajos pendientes. En otras palabras, no son cantidades que, tal y como se han definido, ¡puedan sumarse directamente!

   Por lo tanto, la elección más inteligente consiste en usar el algoritmo IDA* que, además, garantiza encontrar soluciones óptimas.

5. Puesto que los nuevos procesadores se distinguen de los anteriores únicamente en su capacidad de proceso, que de hecho resulta ser p veces mayor que la de los procesadores antiguos, es preciso:

a) Identificar este caracter distintivo en la representación del espacio de problemas:

- Actualizando el espacio de estados. Para ello basta con añadir un nuevo atributo `capacidad` a cada definición de `Procesador` que, por defecto, valdrá 1 y, para los nuevos procesadores será p.
- Modificando los operadores. En este caso es preciso actualizar el tiempo que tarda un procesador en ejecutar un trabajo, de modo que el atributo `t` de la máquina `m` a la que se le asigna un trabajo `t`, será igual a $\texttt{m.t} + \frac{\texttt{t.t}}{\texttt{m.p}}$, en vez de $\texttt{m.t} + \texttt{t.t}$.

b) Actualizar la función heurística sugerida en el tercer apartado, de modo que el reparto de la carga de un trabajo entre varios procesadores debe hacerse considerando el mejor caso —esto es, que esos procesadores ejecutan una parte del trabajo hasta p veces más rápido— resultando:

$$h_4 = \text{máx}\left\{ \min_{m \in \texttt{Procesador}} \left\{ m.\texttt{t} + \frac{1}{pM} \sum_{i=1}^{T'} t_i.\texttt{t} \right\} ; \texttt{makespan} \right\}$$