Departamento de Informática
Universidad Carlos III de Madrid

Grado en Informática

Heurística y Optimización

21 de Enero de 2015

Normas generales del examen

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

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

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

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

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

⑥ No se puede presentar el examen escrito a lápiz

Pregunta 1 (2 puntos)

Se propone alimentar el ganado de una granja con la dieta más económica posible. Dicha dieta debe contener cuatro tipos de nutrientes identificados aquí como A, B, C y D. En el mercado se ofertan tres piensos distintos (P_1, P_2 y P_3) que contienen estos nutrientes. La cantidad, en gramos, de cada componente por kilo de pienso se describe en la siguiente tabla:

	A	B	C	D
P_1	100	–	100	200
P_2	–	100	200	100
P_3	200	100	–	100

La dieta diaria de un animal debe estar compuesta por al menos 0.4 Kg del componente A, 0.6 Kg del componente B, 2Kg del componente C y 1.7 Kg del componente D. El compuesto P_1 cuesta 0.2€/Kg, el compuesto P_2 cuesta 0.08€/Kg y el tercer pienso cuesta 0.25€/Kg.

Se pide responder razonadamente las siguientes preguntas:

(a) **(1 punto)** Modeliza este problema como un problema de programación lineal de modo que su resolución responda a la cuestión ¿Qué cantidades de los piensos P_1, P_2 y P_3 se deben adquirir para que el gasto en comida sea el menor posible?

(b) **(1 punto)** El proveedor de piensos nos explica ahora que como máximo puede servir 60 Kg del compuesto P_1, 30 Kg del compuesto P_2 y 45 Kg del compuesto P_3. Por otra parte, la política comercial de los productores de piensos hace que sea imposible comprar simultaneamente ninguna cantidad de los piensos P_1 y P_2 —esto es, o se compra uno, o se compra el otro, o no se compra ninguno.

Extiende la modelización del primer apartado que considere estas restricciones adicionales. *Indica, únicamente, los cambios sobre la modelización del apartado anterior justificando tus decisiones.*

Pregunta 2 (3 puntos)

Mónica, Adriana, Darío y Roberto deben reunirse uno de los tres primeros días de la semana que viene. La disponibilidad de cada uno se muestra en la tabla siguiente:

	Mónica	Adriana	Darío	Roberto
Lunes	✓	✓		
Martes	✓	✓	✓	✓
Miércoles			✓	✓

Se pide además verificar que los cuatro podrán reunirse una y sólo una vez.

Se pide responder razonadamente las siguientes preguntas:

(a) **(1 punto)** Se pide modelar el problema de decidir qué único día de la semana que viene deben reunirse como un problema de *satisfabilidad lógica*. La fórmula resultante debe expresarse en *Forma Normal Conjuntiva*.

Indica claramente las variables elegidas y su significado así como el propósito de cada cláusula.

(b) **(½ puntos)** ¿La fórmula obtenida en el apartado anterior es satisfacible? Si es así indica un modelo que la satisfaga. Si no es así indica por qué.

Considérese ahora, en su lugar, la organización de una segunda reunión entre las mismas personas en alguno de cuatro días diferentes —que no tiene por qué ser único. La disponibilidad de cada uno se muestra en la tabla siguiente:

	Mónica	Adriana	Darío	Roberto
Lunes		✓		✓
Martes	✓	✓		✓
Miércoles	✓	✓	✓	
Jueves			✓	✓

Se pide responder razonadamente las siguientes preguntas:

(c) **(½ puntos)** Se pide modelar el problema de decidir qué día de la semana que viene deben reunirse como un problema de *satisfacción de restricciones*.

Indica claramente el significado y propósito de cada decisión tomada.

(d) **(½ puntos)** Comprueba si la disponibilidad de Mónica y Adriana es *camino-consistente* respecto de la disponibilidad de Roberto y, después, de Darío.

Si lo es, ¿el problema es resoluble? ¿Si o no? ¿Por qué?

(e) **(½ puntos)** Fuerza la *arco-consistencia* en toda la red de restricciones hasta que no haya cambios en el dominio de ninguna variable.

¿Puede probarse de esta manera si el problema es resoluble o irresoluble? ¿Por qué?

Pregunta 3 (3 puntos)

Considerése el siguiente problema de Programación Lineal:

$$\begin{array}{rcrcr} \text{mín } z = -x_1 - 2x_2 & & & & \\ x_1 & + & x_2 & \leq & 16 \\ x_1 & + & x_2 & = & 6 \\ 3x_1 & + & x_2 & = & 12 \\ & & \mathbf{x} \geq \mathbf{0} & & \end{array}$$

Se pide responder razonadamente las siguientes preguntas:

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

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

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

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

(f) **(½ puntos)** Aplica el algoritmo de *ramificación y acotación* para resolver el mismo problema de programación lineal sujeto a la restricción de que todas las variables de decisión deben tomar valores enteros no negativos.

Pregunta 4 (2 puntos)

Una empresa de paquetería gestiona la entrega de paquetes en varias ciudades. Todas las mañanas llegan a cada ciudad los paquetes que deben distribuirse, y para los que se conoce la dirección de entrega. Las entregas se realizan con furgonetas que recogen en el almacén de cada ciudad un paquete, y sólo uno, y lo entregan en la dirección de destino. Todas las furgonetas son iguales en capacidad y rendimiento, y puede asumirse que hay siempre más furgonetas que paquetes a distribuir. Además, puede asumirse que nunca habrá colisiones entre diferentes furgonetas en la misma ciudad, para el cálculo de cada ruta. Por último, *ninguna furgoneta puede dirigirse a una ciudad diferente que aquella en la que recogió un paquete.*

La empresa está ahora interesada en minimizar la distancia recorrida total en todas las ciudades por todas las furgonetas, desde cada almacén hasta cada punto de entrega.

Se pide responder razonadamente las siguientes preguntas:

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

(b) **(½ puntos)** Habida cuenta que la empresa está interesada en calcular soluciones óptimas, ¿qué algoritmo de búsqueda *no informada* sugerirías para su resolución?

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

Soluciones del examen de Heurística y Optimización Enero 2015

Problema 1

Este problema está inspirado en un ejercicio del software PHPSimplex[1] que, a su vez, está inspirado en el famoso problema de la Dieta de Stigler[2]. En particular, el primer apartado es prácticamente una copia del ejemplo indicado en primer lugar.

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

   Variables de decisión En este problema en particular resulta fácil advertir que son variables de decisión las cantidades que deben comprarse de cada compuesto:

   x_i: Kgs que deben comprarse del compuesto P_i

   Restricciones Las únicas restricciones que hay en la primera parte del problema son relativas a la cantidad mínima de cada nutriente (o componente). En particular, a la vista de la información que hay en la tabla que se muestra en el enunciado y la selección de variables de decisión mostrada más arriba, es fácil observar que $(0{,}1x_1 + 0{,}2x_3)$ es la cantidad del componente A que hay en las cantidades x_1 y x_3 de los piensos P_1 y P_3 respectivamente. Nótese que el segundo pienso no forma parte de esta expresión porque no contiene ninguna cantidad del nutriente A. Como quiera que el enunciado advertía que al menos debía haber 0.4Kg del componente A, la primera restricción es: $0{,}1x_1 + 0{,}2x_3 \geq 0{,}4$.

   Todas las restricciones de la primera parte de este problema se muestran a continuación:

$$\begin{array}{lccccccc}
\text{Componente A:} & 0{,}1x_1 & & & + & 0{,}2x_3 & \geq & 0{,}4 \\
\text{Componente B:} & & + & 0{,}1x_2 & + & 0{,}1x_3 & \geq & 0{,}6 \\
\text{Componente C:} & 0{,}1x_1 & + & 0{,}2x_2 & & & \geq & 2 \\
\text{Componente D:} & 0{,}2x_1 & + & 0{,}1x_2 & + & 0{,}1x_3 & \geq & 1{,}7 \\
& & & \mathbf{x} \geq 0 & & & &
\end{array}$$

   Función objetivo Se trata de un problema de minimización de gastos, de modo que tendremos una función del tipo mín z. En particular, el término que se desea minimizar es el gasto en pienso. Si el primer pienso cuesta 0.2€/Kg entonces la inversión en compra del primer compuesto será igual a $0{,}2x_1$ donde x_1 representa (como se indica más arriba) el número de kilos del primer pienso.

   Razonando análogamente para el segundo y tercer compuesto, resulta la función objetivo:

$$\text{mín } z = 0{,}2x_1 + 0{,}08x_2 + 0{,}25x_3$$

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

$$\begin{array}{rcccccc}
\multicolumn{7}{c}{\text{mín } z = 0{,}2x_1 + 0{,}08x_2 + 0{,}25x_3} \\
0{,}1x_1 & & & + & 0{,}2x_3 & \geq & 0{,}4 \\
& + & 0{,}1x_2 & + & 0{,}1x_3 & \geq & 0{,}6 \\
0{,}1x_1 & + & 0{,}2x_2 & & & \geq & 2 \\
0{,}2x_1 & + & 0{,}1x_2 & + & 0{,}1x_3 & \geq & 1{,}7 \\
\multicolumn{7}{c}{\mathbf{x} \geq 0}
\end{array}$$

Una consideración importante es si el problema es, o no, de programación entera. El enunciado no proporcionaba ninguna información de este tipo y resulta razonable asumir que los kilos de cada pienso que se compren deben ser cantidades enteras. En la modelización sugerida no se ha hecho esta suposición.

[1] http://www.phpsimplex.com/problema_dieta.htm

[2] http://en.wikipedia.org/wiki/Stigler_diet

2. Para resolver este problema es preciso emplear *variables binarias* que representen si se compran o no algunas cantidades del primer y segundo compuesto. Sean estas variables b_1 y b_2 respectivamente:

   b_i: determina si se ha comprado (valor 1) o no (valor 0) alguna cantidad del compuesto P_i

   de modo que la restricción que establece que no es posible comprar los dos compuestos simultaneamente se modela de la manera:

$$b_1 + b_2 \leq 1$$

   Nótese el uso de $\leq$ en vez de $=$, y es que el enunciado sólo prohíbe comprar simultaneamente cantidades positivas de los piensos P_1 y P_2, pero es posible no comprar ninguno de los dos.

   Las variables binarias deben incorporarse ahora a las restricciones del problema de Programación Lineal considerado en el primer apartado —y no a la función objetivo que sólo sirve para medir el gasto de cualquier solución. Por lo tanto, una vez que se ha forzado que sólo una de las variables binarias como máximo puede tomar el valor 1, es preciso modelizar la disponibilidad máxima de cada compuesto como se indica a continuación:

$$\begin{array}{rrrcr} x_1 & & & \leq & 60b_1 \\ & x_2 & & \leq & 30b_2 \\ & & x_3 & \leq & 45 \end{array}$$

   o, análogamente, si se desean separar los términos independientes del uso de variables de decisión:

$$\begin{array}{rrrrrcr} x_1 & & - & 60b_1 & & \leq & 0 \\ & x_2 & & & - \; 30b_2 & \leq & 0 \\ & & x_3 & & & \leq & 45 \end{array}$$

   De modo que resulta ahora el problema de Programación Lineal mostrado a continuación:

$$\text{mín } z = 0{,}2x_1 + 0{,}08x_2 + 0{,}25x_3$$

$$\begin{array}{rrrrrrrrrcr} 0{,}1x_1 & & & + & 0{,}2x_3 & & & & & \geq & 0{,}4 \\ & + & 0{,}1x_2 & + & 0{,}1x_3 & & & & & \geq & 0{,}6 \\ 0{,}1x_1 & + & 0{,}2x_2 & & & & & & & \geq & 2 \\ 0{,}2x_1 & + & 0{,}1x_2 & + & 0{,}1x_3 & & & & & \geq & 1{,}7 \\ x_1 & & & & & - & 60b_1 & & & \leq & 0 \\ & & x_2 & & & & & - & 30b_2 & \leq & 0 \\ & & & + & x_3 & & & & & \leq & 45 \end{array}$$

$$\mathbf{x} \geq 0, b_1, b_2 \in \{0, 1\}$$

Problema 2

Antes de presentar la resolución de este problema, se advierte que difícilmente se emplearía una técnica (*satisfabilidad lógica*) o la otra (*satisfacción de restricciones*) para resolver un problema como éste. El motivo es que el único ejercicio computacional que debe hacerse para resolver la fecha en que es posible convocar a varias personas *habida cuenta de la disponibilidad de cada participante* consiste en comprobar si hay un día en el que pueden todos los participantes. Si n es el número de convocados y m el número de días posibles, este problema puede resolverse en $O(nm)$.

El problema sí es interesante, sin embargo, en otras circunstancias. Por ejemplo, si la disponibilidad de cada individuo debe decidirse también automáticamente. Para no hacer el problema demasiado complicado, cualquier otra alternativa se descartó intencionadamente.

1. En satisfabilidad lógica, las variables sólo pueden tomar los valores cierto ($\top$) o falso ($\bot$). Se trata, por lo tanto, de variables *booleanas*.

   Se propone el uso de dos tipos de variables: p_i y q_j. Las primeras se usan para decidir automáticamente, qué día debe celebrarse la reunión. Las segundas sirven para decidir quién asiste a la reunión.

   Por lo tanto, se definen las variables:

   p_i: La reunión se celebra el día i (i = Lunes, Martes, Miércoles)
   q_j: La persona j asiste a la reunión

   de donde resultan hasta 7 variables: 3 en el primer grupo (p_L, p_M y p_X para los días Lunes, Martes y Miércoles respectivamente) y cuatro en el segundo —q_M, q_A, q_D y q_R para Mónica, Adriana, Darío y Roberto respectivamente.

   Las cláusulas servirán para modelizar las restricciones del problema:

   a) Antes que nada, es preciso asegurarse que todas las personas podrán participar en la reunión: $q_M \wedge q_A \wedge q_D \wedge q_R$.
   En total, cuatro cláusulas unitarias.

   b) En segundo lugar, se modelizan las personas que podrían asistir a la reunión si se hace un día determinado.
   Por ejemplo, si la reunión se hace el Lunes, entonces sólo podrían asistir Mónica y Adriana. Esta relación se representa en lógica proposicional como sigue:

   $$p_L \rightarrow (q_M \wedge q_A \wedge \overline{q_D} \wedge \overline{q_R})$$

   que es equivalente a:

   $$(\overline{p_L} \vee (q_M \wedge q_A \wedge \overline{q_D} \wedge \overline{q_R}))$$
   $$(\overline{p_L} \vee q_M) \wedge (\overline{p_L} \vee q_A) \wedge (\overline{p_L} \vee \overline{q_D}) \wedge (\overline{p_L} \vee \overline{q_R})$$

   de donde han salido exactamente cuatro cláusulas de longitud 2. Nótese que en la modelización de esta implicación también se debe indicar, explícitamente, quién no podría asistir a la reunión en caso de que se celebrara el Lunes.
   Si la reunión se celebra el Martes, la expresión resultante es:

   $$p_M \rightarrow (q_M \wedge q_A \wedge q_D \wedge q_R)$$

   de donde resultan ahora cuatro cláusulas de longitud 2 operando como se indica a continuación:

   $$(\overline{p_M} \vee (q_M \wedge q_A \wedge q_D \wedge q_R))$$
   $$(\overline{p_M} \vee q_M) \wedge (\overline{p_M} \vee q_A) \wedge (\overline{p_M} \vee q_D) \wedge (\overline{p_M} \vee q_R)$$

   Operando análogamente, se obtendrían cuatro cláusulas para el caso en el que la reunión se celebrara el Miércoles:

   $$(\overline{p_X} \vee \overline{q_M}) \wedge (\overline{p_X} \vee \overline{q_A}) \wedge (\overline{p_X} \vee q_D) \wedge (\overline{p_X} \vee q_R)$$

   En total, han resultado 12 cláusulas.

   c) Es preciso modelar, además, que la reunión debería poder celebrarse alguno de los tres días.
   De esta consideración resulta una única cláusula de longitud 3: $(p_L \vee p_M \vee p_X)$

d) Por último, aunque se trata de una restricción más bien *artficial* el enunciado advertía que la reunión se debía poder celebrar uno y sólo un día.

Por lo tanto, si la reunión se celebra el Lunes se desea comprobar que no es posible celebrarla ni el Martes, ni el Miércoles:

$$p_L \rightarrow (\overline{p_M} \wedge \overline{p_X})$$

Como en el caso del primer grupo de cláusulas, esta implicación se puede representar automáticamente en Forma Normal Conjuntiva transformando primero la implicación:

$$\overline{p_L} \vee (\overline{p_M} \wedge \overline{p_X})$$

que es estrictamente equivalente a la expresión:

$$(\overline{p_L} \vee \overline{p_M}) \wedge (\overline{p_L} \vee \overline{p_X})$$

De la misma manera, si la reunión se celebra el Martes se debe comprobar que no es posible hacerlo ni el Lunes ni el Miércoles:

$$(\overline{p_M} \vee \overline{p_L}) \wedge (\overline{p_M} \vee \overline{p_X})$$

El último grupo de cláusulas es:

$$(\overline{p_X} \vee \overline{p_L}) \wedge (\overline{p_X} \vee \overline{p_M})$$

De donde resultan, en total, sólo tres cláusulas diferentes (puesto que hay otras tres que se repiten en las diferentes expresiones).

La fórmula de lógica proposicional solicitada es, por lo tanto, la que resulta de hacer la conjunción de todas las cláusulas calculadas en los pasos anteriores. En total, la fórmula consta de 7 variables y 20 cláusulas.

Nótese que no ha sido necesario en absoluto modelizar la disponibilidad de cada persona para cada día. Aunque esa es otra modelización posible, es peor puesto que habría, en total, hasta nm variables (donde n y m están definidas justo antes de la solución del primer apartado más arriba). En este caso, habría en total hasta 12 variables que es significativamente mayor que las 7 obtenidas con la modelización anterior. Esta distinción es importante porque si la expresión resultante se fuera a resolver con Davis-Putnam o Davis-Putnam-Logemann-Loveland, es precisamente el número de variables el que representa el número de pasos necesarios y por lo tanto, se prefiere reducir el número de variables.

2. ¡Obviamente sí! Esto puede determinarse simplemente por inspección de la tabla mostrada en el enunciado. En particular, la reunión puede celebrarse el Martes y se verifica, además, que no es posible celebrarla ni el Lunes ni el Miércoles.

   Esta solución se representa con el modelo siguiente:

$$M = \{p_L = \bot, p_M = \top, p_X = \bot, q_M = \top, q_A = \top, q_D = \top, q_R = \top\}$$

3. 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. A continuación se presenta una modelización de cada una de estas partes:

Variables Se propone el uso de *variables* que representen el día en que puede reunirse cada persona. Hay, por lo tanto, hasta cuatro variables (una por cada persona representada aquí genéricamente como i):

x_i: Disponibilidad de la persona i

de modo que $X = \{x_M, x_A, x_D, x_R\}$

Dominios El dominio de cada variable es el conjunto {Lunes, Martes, Miércoles, Jueves} esto es, simplemente los días considerados en los que podría celebrarse la reunión.

Por lo tanto, $D_i = \{\text{Lunes}, \text{Martes}, \text{Miercoles}, \text{Jueves}\}$, y como esto es cierto para todas las variables identificadas en el paso anterior, se tiene que $D = \{D_M, D_A, D_D, D_R\}$

Otra alternativa, mejor aún, consiste en eliminar anticipadamente de cada dominio los días en los que la persona correspondiente no podría reunirse. De esta manera:

D_M: {Martes, Miércoles}
D_A : {Lunes, Martes, Miércoles}
D_D : {Miércoles, Jueves}
D_R : {Lunes, Martes, Jueves}

En lo sucesivo, se trabajará de hecho con esta segunda modelización de los dominios.

Restricciones Las restricciones modelan los valores simultaneamente legales para cada par de variables. Para su cálculo es preciso tener en cuenta las restricciones del problema que, en esta parte, consisten únicamente en asegurarse que todas las personas puedan coincidir un día determinado. Por este motivo, las restricciones se calculan trivialmente como tuplas que tienen siempre los mismos valores y se muestran a continuación (donde R_{ij} representa las restricciones entre las variables x_i y x_j):

R_{MA}	{Martes, Martes}, {Miércoles, Miércoles}
R_{MD}	{Miércoles, Miércoles}
R_{MR}	{Martes, Martes}
R_{AD}	{Miércoles, Miércoles}
R_{AR}	{Lunes, Lunes}, {Martes, Martes}
R_{DR}	{Jueves, Jueves}

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

   Primero se estudia la camino consistencia de R_{MA} respecto de $x_k = x_R$. Por lo tanto, las restricciones involucradas son R_{MA}, R_{MR} y R_{AR}:

R_{MA}	{Martes, Martes}, {Miércoles, Miércoles}
R_{MR}	{Martes, Martes}
R_{AR}	{Lunes, Lunes}, {Martes, Martes}

   Si se considera la primera tupla de R_{MA} : (Martes, Martes), se observa que hay un valor en el dominio de la tercera variable, x_R = Martes tal que la tupla (Martes, Martes) aparece también en R_{MR} y R_{AR}. Este no es el caso, sin embargo, para la segunda tupla de R_{MA}, (Mircoles, Mircoles) puesto que ni R_{MR} ni R_{AR} contienen tuplas con el Miércoles. Por lo tanto, la segunda tupla de R_{MA} puede eliminarse y como esta restricción no es vacía se concluye que x_M y x_A son camino consistentes respecto de x_R.

   Ahora se considera el estudio de la camino consistencia de R_{MA} respecto de $x_k = x_D$. Ahora, las restricciones involucradas son R_{MA}, R_{MD} y R_{AD}:

R_{MA}	{Martes, Martes}, {Miércoles, Miércoles}
R_{MD}	{Miércoles, Miércoles}
R_{AD}	{Miércoles, Miércoles}

Operando como antes es muy fácil observar que ahora puede eliminarse la primera tupla de R_{MA}, puesto que no hay ninguna tupla en las otras dos restricciones que contengan el Martes. Como la segunda tupla de R_{MA} sí supera la camino consistencia respecto de x_D, se tiene que las variables x_M y x_A son camino consistentes respecto de x_D.

En respuesta a la última pregunta de este apartado, nótese como la camino consistencia de R_{MA} respecto de las otras dos variables no sirve, en absoluto, para garantizar que el problema sea resoluble. De hecho, una simple inspección de la tabla considerada en este apartado revela que no hay ningún día en el que puedan reunirse las cuatro personas convocadas[3].

En términos generales, la camino consistencia de longitud k no sirve para garantizar que un problema con $(k+1)$ variables sea resoluble.

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

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

   En este problema, no hay un par de variables que sean arco-consistentes. En particular, si $x_M =$ Martes, no hay ningún valor v en el dominio de x_D ($v \in D_D$) tal que $(\text{Martes}, v) \in R_{MD}$, de modo que Martes debería eliminarse del dominio de Mónica, D_M. Después de eliminar el Martes del dominio D_M, también debería eliminarse de los dominios de Adriana y Roberto, puesto que ya no habría tuplas en sus restricciones respectivas que incluyan el valor Martes para x_M.

   En general, para cualquier día hay una variable x_i que no tiene definido ese valor en su dominio y, por lo tanto, lo elimina del resto de variables a la hora de forzar la arco-consistencia. Actuando de esta manera, el resultado de forzar la arco consistencia en toda la red de restricciones resulta en dominios vacíos para todas las variables ($D_i = \varnothing$) y, por lo tanto, a la detección automática de que el problema es irresoluble.

Problema 3

Este problema es el ejercicio 3.5.b) del libro Sixto Ríos Insua. Investigación Operativa - Optimización. Editorial Ramón Areces. 1988 (página 101) que se incluye en la bibliografía del curso. Se trata de un problema extremadamente sencillo, el motivo es que la segunda restricción del enunciado ($x_1 + x_2 = 6$) domina estrictamente a la primera ($x_1 + x_2 \leq 16$) o, en otras palabras, cualquier par de valores que verifique la segunda restricción verifica la primera. Por lo tanto, la primera podría quitarse directamente del problema de Programación Lineal resultando entonces un problema de dos variables con dos ecuaciones. Este problema es trivial porque tiene solución única, es decir, ¡no hay nada que optimizar!

En las soluciones que siguen, sin embargo, se resuelve el problema con las técnicas vistas en clase con el único propósito de ejercitarlas. Como se verá, se alcanzarán las mismas soluciones que si se hubiesen seguido los pasos indicados aquí.

1. Puesto que sólo hay dos variables de decisión, x_1 y x_2, las restricciones se pueden dibujar sobre un plano bidimensional.

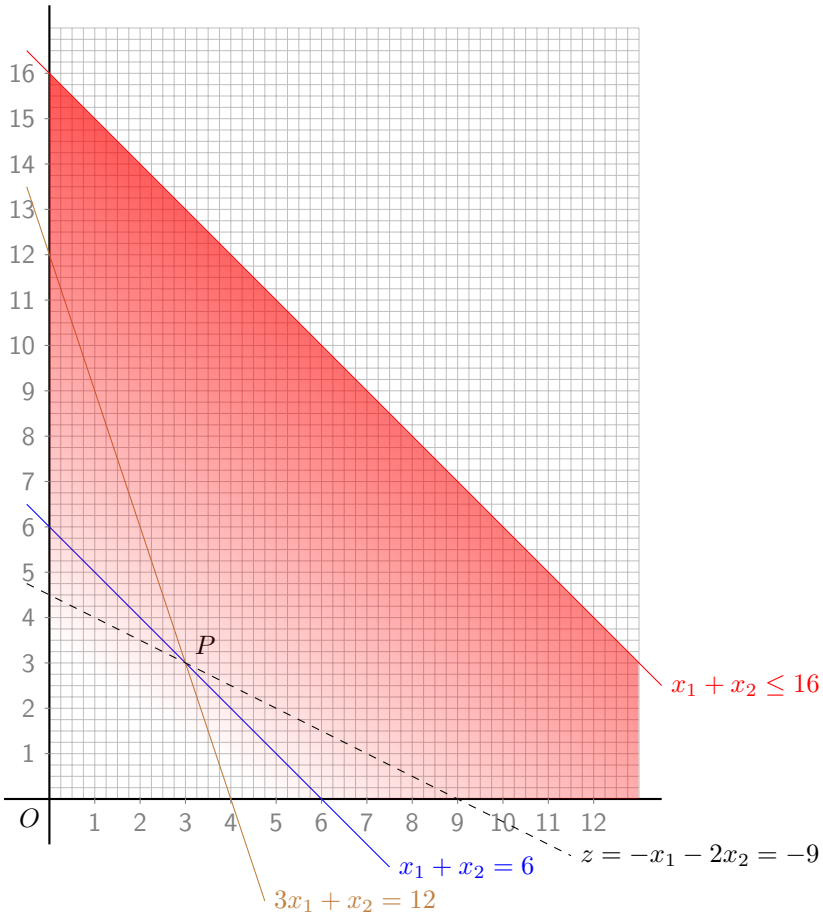
   La siguiente figura muestra gráficamente cada una de las restricciones. Puesto que la segunda y tercera restricción son igualdades, sus regiones consisten únicamente en una línea. Por lo tanto:

[3] Nótese, sin embargo, que en este apartado se ha ejercitado una forma de consistencia entre dos variables específicas respecto de una tercera muy concreta. Si se hubiera forzado la camino consistencia en toda la red de restricciones sí se hubiera detectado que el problema es irresoluble. Éste será, precisamente, el objetivo del siguiente apartado con una forma más sencilla de consistencia, la arco-consistencia

a) La región factible, que consiste en la intersección de las regiones que satisfacen cada restricción resulta únicamente de considerar la intersección de dos líneas. Por lo tanto, la región factible está constituída únicamente por el punto P mostrado en la figura.

b) Tal y como se advertía anteriormente, la primera restricción está dominada por las otras. De hecho, la región que satisface la primera restricción (mostrada en color rojo) contiene el punto P —y esta es una buena forma de reconocer gráficamente la dominancia entre restricciones.

Puesto que la región factible consiste en un único punto, sólo puede haber una curva de isocoste (llamada así porque la función objetivo es de minimización). La curva de isocoste mostrada en la figura indica que el valor de la función objetivo es -9.



Por lo tanto, con una región factible que únicamente contiene un punto, la solución es única y consiste en el punto $(3, 3)$. El valor de la función objetivo en ese punto es -9.

2. En el segundo 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 y, también, que la función objetivo es de minimización. Conviene aquí recordar:

   - Una restricción de la forma $\leq$ está acotada superiormente. Puesto que ninguna variable de decisión puede tomar valores negativos, es preciso *sumar* una *variable de holgura* para forzar la igualdad.

- Análogamente, las restricciones de la forma $\geq$ 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:

$$\begin{array}{c} \text{máx } z = x_1 + 2x_2 \\ \begin{array}{rcrcrcr} x_1 & + & x_2 & + & x_3 & = & 16 \\ x_1 & + & x_2 & & & = & 6 \\ 3x_1 & + & x_2 & & & = & 12 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

La función objetivo se ha convertido a una forma de maximización simplemente multiplicándola por -1[4]. Además, la primera restricción se ha convertido en una forma de igualdad sumando una variable de holgura, x_3. como se discutía en el primer caso mostrado más arriba.

3. La matriz de coeficientes tecnológicos (denotada normalmente como A) del problema de Programación Lineal del apartado anterior no contiene la matriz identidad pero sí contiene una de sus columnas. En particular, el vector columna a_3 es la primera columna de la matriz identidad de la dimensión requerida en este caso, 3, y representada como I_3.

   Para *forzar* la aparición de I_3 en la matriz de coeficientes tecnológicos simplemente se añaden tantas *variables artificiales* como haga falta a las restricciones del problema. Como quiera que estas variables no tienen ningún significado en el problema original, se penalizan con $-\infty$ en la función objetivo.

   De esta manera, resulta ahora el nuevo problema de Programación Lineal:

$$\begin{array}{c} \text{máx } z = x_1 + 2x_2 - \infty x_4 - \infty x_5 \\ \begin{array}{rcrcrcrcrcr} x_1 & + & x_2 & + & x_3 & & & & & = & 16 \\ x_1 & + & x_2 & & & + & x_4 & & & = & 6 \\ 3x_1 & + & x_2 & & & & & + & x_5 & = & 12 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

   que ahora sí contiene I_3 y que está formada por los vectores columna a_3, a_4 y a_5.

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

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

   Paso 0 Cálculo de una solución factible inicial

[4]Una buena práctica consistiría en renombrar la función objetivo ahora como z' para distinguirla de la función objetivo inicial. Aquí se ha utilizado el mismo nombre, z, sin riesgo de cometer ambigüedades.

a) Cálculo de las variables básicas
La primera iteración se inicia con una base igual a la matriz identidad de dimensión 3, tal y como se calculó ya en el apartado anterior. Por lo tanto, son variables básicas en este paso $\{x_3, x_4, x_5\}$

$$B_0 = I_3 \qquad B_0^{-1} = I_3$$
$$x_0^* = B_0^{-1}b = b = \begin{pmatrix} 16 \\ 6 \\ 12 \end{pmatrix} \quad z_0^* = c_{B_0}^T x_0^* = \begin{pmatrix} 0 & -\infty & -\infty \end{pmatrix} \begin{pmatrix} 16 \\ 6 \\ 12 \end{pmatrix} = -18\infty$$

Por supuesto, $-18\infty = -\infty$. En realidad, ∞ se usa aquí simbólicamente como una variable que puede sustituirse por un valor estrictamente positivo arbitrariamente grande. Por lo tanto, se preservan los coeficientes para facilitar comparaciones entre expresiones que lo contengan.

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 (aunque en una iteración con una base igual a la matriz identidad, $y_i = a_i$):

$$z_1 - c_1 = c_{B_0}^T B_0^{-1} a_1 - c_1 = \begin{pmatrix} 0 & -\infty & -\infty \end{pmatrix} I_3 \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} - 1 = -4\infty - 1$$
$$z_2 - c_2 = c_{B_0}^T B_0^{-1} a_2 - c_2 = \begin{pmatrix} 0 & -\infty & -\infty \end{pmatrix} I_3 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - 2 = -2\infty - 2$$

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

$$\text{mín} \left\{ \frac{16}{1}, \frac{6}{1}, \frac{12}{3} \right\}$$

y sale la variable x_5 que es la que se corresponde con la tercera fracción (puesto que ocupa la tercera posición en la base), precisamente una de las variables artificiales, como debía esperarse.

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 $\{x_1, x_3, x_4\}$

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

$$x_1^* = B_1^{-1}b = \begin{pmatrix} 4 \\ 12 \\ 2 \end{pmatrix} \quad z_1^* = c_{B_1}^T x_1^* = \begin{pmatrix} 1 & 0 & -\infty \end{pmatrix} \begin{pmatrix} 4 \\ 12 \\ 2 \end{pmatrix} = 4 - 2\infty$$

Como $4 - 2\infty > -18\infty$ para cualquier sustitución del símbolo ∞ por una variable estrictamente positiva arbitrariamente grande, se confirma que el valor de la función objetivo ha crecido entre la iteración anterior y ésta.

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. Por lo tanto, sólo queda una variable no básica por considerar, x_2. Aunque sea sólo una, es preciso hacerlo: si su coste reducido fuera positivo, SIMPLEX habría acabado; en otro caso sería preciso calcular, al menos, una iteración más.

$$z_2 - c_2 = c_{B_1}^T B_1^{-1} a_2 - c_2 = \begin{pmatrix} 1 & 0 & -\infty \end{pmatrix} \begin{pmatrix} 0 & 0 & \frac{1}{3} \\ 1 & 0 & -\frac{1}{3} \\ 0 & 1 & -\frac{1}{3} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - 2 = -\frac{1}{3}(2\infty + 5)$$

y la variable x_2 debe entrar en la base puesto que tiene un coste reducido estrictamente negativo.

c) Selección de la variable de salida
Como siempre, la variable de salida se calcula en atención al mínimo cociente x_1/y_{ij} donde x_i es la variable elegida en el paso anterior para añadirse a la base (x_2) e y_{ij} son las componentes de su vector $\mathbf{y}$ también calculado en el paso anterior:

$$\text{mín} \left\{ \frac{4}{\frac{1}{3}}, \frac{12}{\frac{2}{3}}, \frac{2}{\frac{2}{3}} \right\}$$

y sale la variable x_4 que es la que se corresponde con la tercera fracción. Una vez más, sale a continuación la segunda variable artificial básica, como debía esperarse.

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

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

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

$$x_2^* = B_2^{-1}b = \begin{pmatrix} 3 \\ 3 \\ 10 \end{pmatrix} \quad z_2^* = c_{B_2}^T x_2^* = \begin{pmatrix} 1 & 2 & 0 \end{pmatrix} \begin{pmatrix} 3 \\ 3 \\ 10 \end{pmatrix} = 9$$

b) Selección de la variable de entrada
Razonando como antes, no es preciso calcular el coste reducido de ninguna variable artificial no básica (puesto que es imposible que den valores negativos). Como las variables no básicas son todas artificiales, SIMPLEX concluye en esta iteración con la solución calculada en el punto anterior.

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

$$x^* = \begin{pmatrix} 3 \\ 3 \\ 10 \\ 0 \\ 0 \end{pmatrix} \quad z^* = 9$$

A propósito de la advertencia del comienzo de la solución de este problema, el resultado se podía haber anticipado eliminando simplemente la primera restricción. En ese caso, resulta simplemente un sistema algebraico de dos variables con dos ecuaciones cuya matriz de coeficientes es:

$$A = \begin{pmatrix} 1 & 1 \\ 3 & 1 \end{pmatrix}$$

y que se resuelve simplemente multiplicando la inversa de esta matriz por el vector columna de términos independientes:

$$\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} 6 \\ 12 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

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

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

- El problema es factible porque la solución no contiene valores positivos para ninguna variable artificial.
- La solución es única porque los costes reducidos de la última iteración son todos estrictamente positivos. Eso significa que cualquier cambio en la base implicaría un decremento neto en el valor de la función objetivo.
  En particular, conviene advertir aquí que realmente no había ningún problema de optimización puesto que la región factible consiste en un único punto como se advertía en la resolución del primer apartado. De hecho, la única variable de holgura que hay toma un valor estrictamente positivo pero su importancia es nula puesto que representa la cantidad que debe sumarse para verificar una restricción (la primera) que está dominada por otra, la segunda. De hecho, es muy fácil verificar que si el recurso de la primera restricción decrece una cantidad menor o igual que 10 unidades, la solución sigue siendo exactamente la misma.
- El valor de la función objetivo está acotado porque siempre se pudo aplicar la regla de salida con denominadores que siempre fueron estrictamente positivos.

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

6. El algoritmo de ramificación y acotación establece que primero se inicializa la variable B a $+\infty$ (puesto que esta se usa para calcular el mínimo de una serie de números) y, a continuación se resuelve el problema original ignorando las restricciones de integridad.

   Puesto que la solución obtenida es entera, el problema *relajado* tiene solución entera y, por lo tanto, se actualiza el valor del umbral B al valor de la función objetivo de la solución óptima. Como no hay ningún otro subproblema pendiente de resolverse, el algoritmo de ramificación y acotación concluye en la primera iteración con la solución encontrada en el apartado 4.

Problema 4

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

   Estados Un estado en este problema está caracterizado por información de cada ciudad, la lista de entregas que deben realizarse en cada ciudad y, por último, el estado de todas las furgonetas disponibles para los diferentes servicios. De hecho, cada entrega estará cualificada con información de la furgoneta que hace el servicio correspondiente:

   Entregas Cada entrega está identificada con un código de la ciudad a la que llega y donde debe servirse (`ciudad`). Además, contendrá información de la dirección de entrega (`direccion`) y un código único que sirva para distinguirla de otras entregas, `id`.
   Para poder gestionar la actividad diaria de la empresa de paquetería considerada en el enunciado, cada entrega debe asignarse a un vehículo (`furgoneta`) del que se mantendrá información adicional como se explica a continuación.

   Furgonetas Cada furgoneta está identificada con un código único que la distingue de las demás (`furgoneta`). Además, contendrá información de su posición actual distinguida con dos coordenadas que la localizarán perfectamente sobre el mapa de la ciudad (`xpos`, `ypos`) y el identificador del paquete que transporta, `paquete`. Por último, para habilitar la aplicación de algoritmos de búsqueda, debe contener un campo de distancia recorrida (`distancia`) que se actualizará en la expansión de cada nodo (con el operador `Mover` descrito más abajo).

   Mapa Para poder determinar las rutas óptimas para cada furgoneta, el sistema debe conocer la topografía de cada ciudad que se representará con un mapa único con información de todas las ciudades consideradas por la empresa de paquetería. Cada mapa consiste simplemente en un par de coordenadas (`xpos`, `ypos`) que apuntan a los pares de coordenadas adyacentes a él.

   Si una furgoneta no tiene ningún paquete asignado, su identificador de paquete (`paquete`) será nulo. Análogamente, si una entrega no tiene asignado aún un vehículo, su referencia `furgoneta` será también nulo.

   Operadores En la definición de operadores es importante describir sus precondiciones/postcondiciones y, además, el coste que tienen. En este problema pueden distinguirse dos operadores: el primero consiste en asignar furgonetas a servicios a primera hora de la mañana; el segundo consiste en determinar los movimientos de la furgoneta en cada ciudad de modo que haga el recorrido óptimo hasta su entrega.

   Asignar Puesto que el enunciado advierte que hay una cantidad arbitrariamente grande de furgonetas, la asignación se hará una única vez por la mañana, de modo que la única precondición que debe verificarse es que se trate de una furgoneta disponible y una entrega sin furgoneta asignada. La postcondición es, simplemente, que el paquete asignado se escribe en el atributo `paquete` de la furgoneta escogida.

Puesto que el objetivo del problema es minimizar la distancia total recorrida por todas las furgonetas en todas las ciudades, esta operación no afecta a ese coste (puede asumirse libremente que las furgonetas están todas estacionadas por la mañana junto al almacén, de modo que no hay que hacer ningún recorrido para recoger un paquete) y, por lo tanto, tendrá coste 0.

Mover El movimiento de cada furgoneta se decide en base a la información disponible en el mapa. Para cualquier posición determinada por el par (`xpos`, `ypos`) es posible moverse a cualquier posición apuntada por élla en el mapa, o sea, que sea adyacente a élla.
Es una precondición de este operador que tenga asignado un paquete mediante la ejecución del operador `Mover` (esto es, ¡no queremos que el algoritmo mueva vehículos que no transportan nada!). La postcondición de este operador consiste en sobreescribir el contenido de sus atributos `xpos` e `ypos` con la nueva posición, y actualizar la información de la distancia recorrida por esta furgoneta.
El coste del operador es igual a la distancia recorrida con su aplicación.

2. Para minimizar la distancia recorrida en total por todas las furgonetas y habida cuenta de que ninguna furgoneta puede dirigirse a una ciudad diferente de aquella donde recogió un paquete, basta con encontrar las rutas óptimas para cada furgoneta en cada servicio. Esto es, el *enrutamiento* de cada furgoneta es independiente de otras, no sólo si son de ciudades diferentes, ¡sino incluso si son de la misma ciudad! Lo último es cierto considerando (como advertía el enunciado) que no hay colisiones entre las furgonetas de una misma ciudad[5].

   Por lo tanto, se trata de resolver un problema de *minimización* en un grafo de estados donde los operadores tienen costes arbitrarios. Los algoritmos estudiados con este propósito son fundamentalmente dos:

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

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

   En particular, es lógico esperar que en una ciudad haya muchas transposiciones puesto que es posible llegar al mismo sitio de muchas maneras diferentes. Por lo tanto, inicialmente se elige el algoritmo Dijkstra. Ahora bien, puesto que la distancia total mínima resulta de calcular rutas óptimas para cada servicio, se propone aplicar tantos Dijkstras como servicios haya. La solución global consiste en la concatenación de todos los servicios —un algoritmo Dijkstra que considerara simultáneamente el movimiento de varias furgonetas incurriría, con toda seguridad, en un consumo de memoria absolutamente prohibitivo.

   Si aún así, la memoria consumida por Dijkstra excediera la disponible, entonces se sugiere el uso de DFBnB, uno por cada servicio que debe realizarse.

3. Para el diseño de una función heurística *admisible* y bien informada, se sugiere el uso de la técnica de *relajación de restricciones*. Habida cuenta de que cada algoritmo de búsqueda sólo resolverá el *enrutamiento* de cada furgoneta por separado, se propone una relajación que afecta a una única furgoneta.

   En el proceso de encontrar el camino óptimo entre dos puntos n y t[6], respectivamente, se sugiere relajar la condición (enuncidada en el primer apartado) de que el movimiento de las furgonetas se hace

[5] En caso contrario, se tendría el problema conocido como Multi-Agent Path Finding (MAPF).
[6] Del inglés, *target*.

necesariamente a posiciones adyacentes. El *problema relajado* resultante puede resolverse óptimamente simplemente asumiendo que el movimiento de cada furgoneta se hace a lo largo de una ruta mínima incluso si los puntos intermedios no son adyacentes. La ruta óptima del problema relajado es precisamente una línea recta que une a la furgoneta con su posición de destino, t a partir de su posición original, n. Por lo tanto, la función heurística propuesta es precisamente la *distancia aérea* entre la posición actual de la furgoneta y la posición de destino:

$$h(n) = \sqrt{(n.\texttt{xpos} - t.\texttt{xpos})^2 + (n.\texttt{ypos} - t.\texttt{ypos})^2}$$

donde (como se señalaba en el apartado anterior), la solución óptima global resulta de resolver la aplicación de diferentes algoritmos de búsqueda *heurística*, cada uno de ellos guiado por la función indicada aquí.