Intel.ligència Artificial

Enginyeria en Informàtica

COL.LECCIÓ DE PROBLEMES SOLUCIONS

Departament de Llenguatges i Sistemes Informàtics

UPC FIB Facultat d'Informàtica de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

CURS 2010/2011 2Q

Índice general

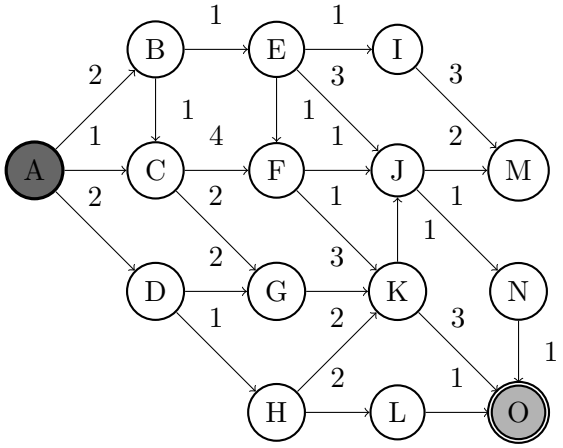
En l'elaboració de les solucions de la col.lecció de problemes d'IA han participat:

Badia Orive, Esteve
Cami Muntane, Jordi
Iglesias Sanchez, Patricia
Lopez Cervilla, Javier
Padrol Sureda, Arnau
Ramos Garcia, Claudia
Tripiana Montes, Carlos
Tur Moreno, David

Responsable de la publicació: Javier Béjar (bejar@lsi.upc.edu)

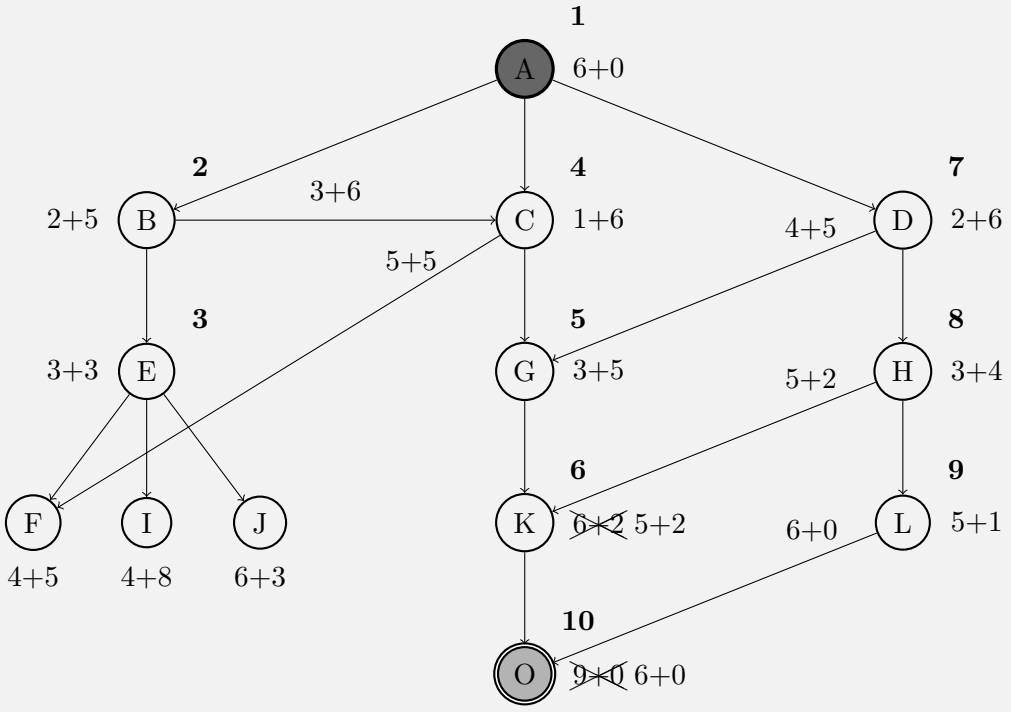
1.1 Problemas solucionados

7. Dado el siguiente grafo donde cada arco indica su coste y la tabla que indica la estimación del coste h hasta la solución, indica cual sería el árbol de búsqueda que se obtendría mediante el algoritmo de A* e IDA* para encontrar el camino entre el nodo A y el nodo O. Haz la generación de los nodos siguiendo el orden alfabético e indica claramente las reprticiones de los nodos y los cambios de coste que aparezcan. ¿Es la función heurística admisible?

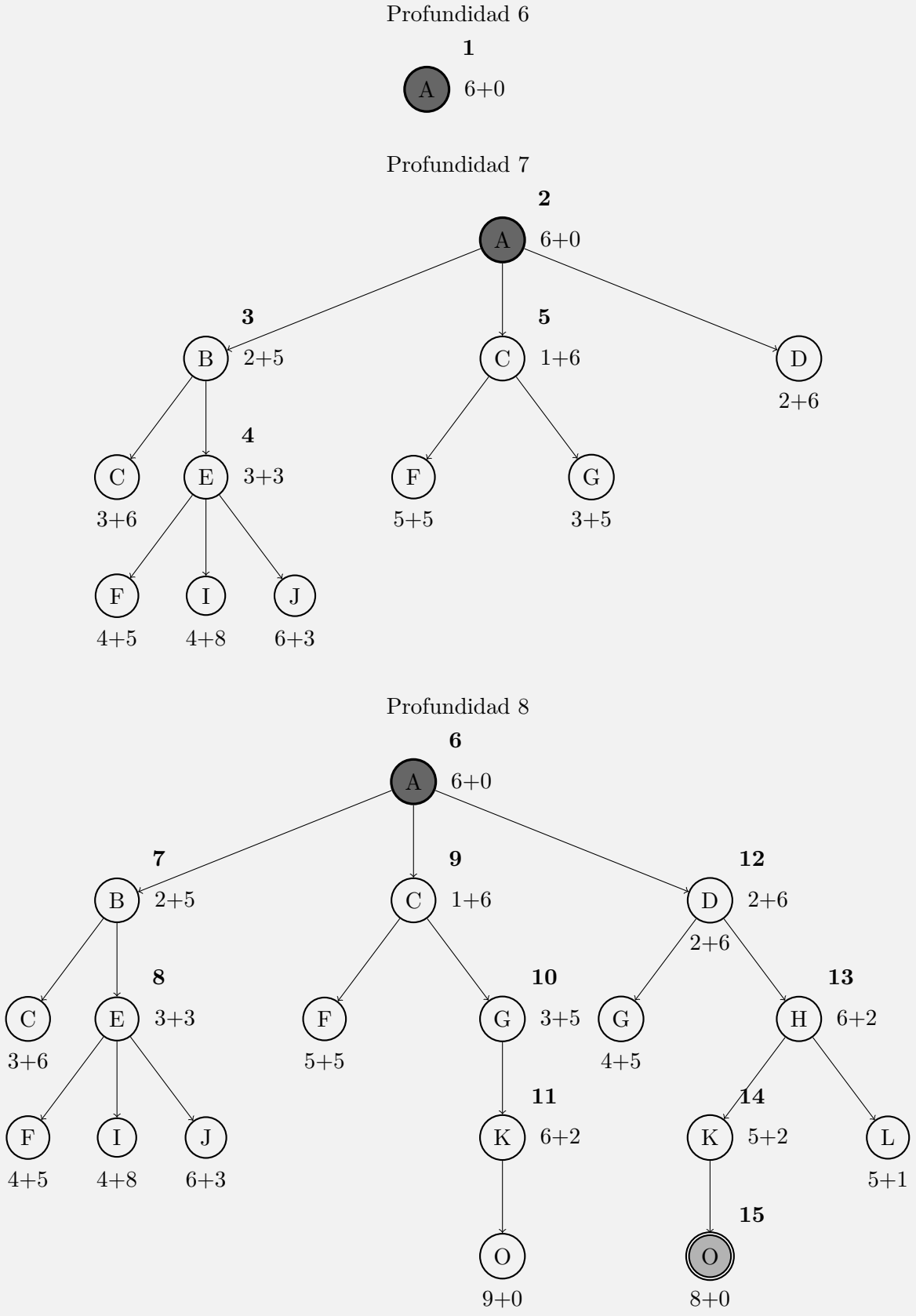


Nodo	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
h(nodo)	6	5	6	6	3	5	5	4	8	3	2	1	5	1	0

Árbol de búsqueda con A*

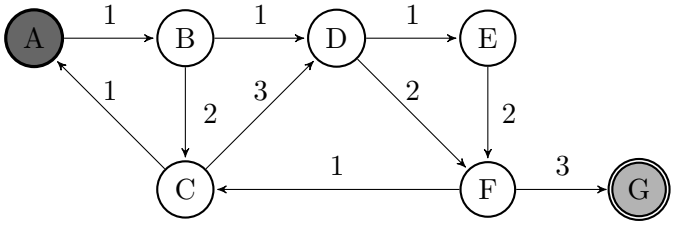


Árbol de búsqueda con IDA*



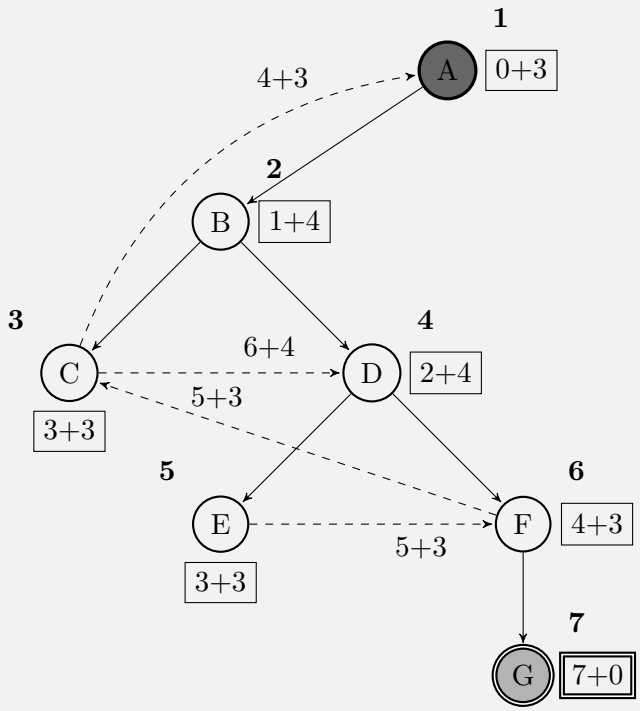
La función heurística no es admisible ya que en varias ocasiones el coste estimado es mayor que el coste real.

14. Dado el siguiente grafo donde cada arco indica su coste y la tabla que indica la estimación del coste h hasta la solución, indica cual sería el árbol de búsqueda que se obtendría mediante el algoritmo de A* e IDA* para encontrar el camino entre el nodo A y el nodo G. Haz la generación de los nodos siguiendo el orden alfabético e indica claramente las reexpansiones de los nodos y los cambios de coste que aparezcan. ¿Es la función heurística admisible?



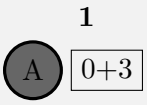
Nodo	A	B	C	D	E	F	G
h(nodo)	3	4	3	4	3	3	0

Árbol de búsqueda con A*

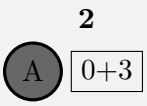


Árbol de búsqueda con IDA*

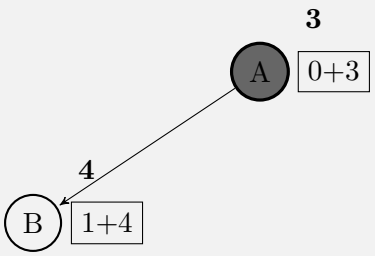
Profundidad 3



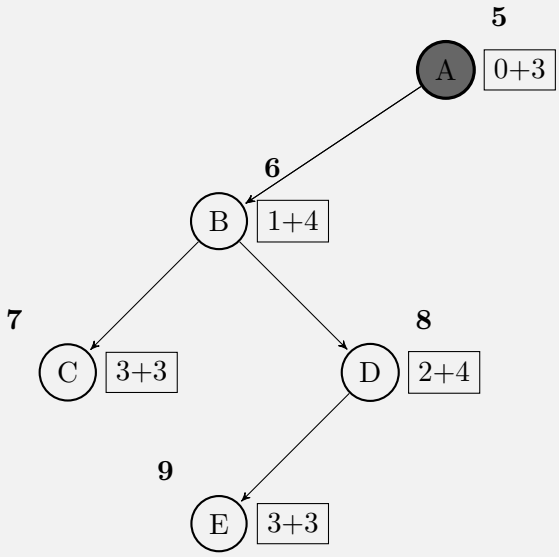
Profundidad 4



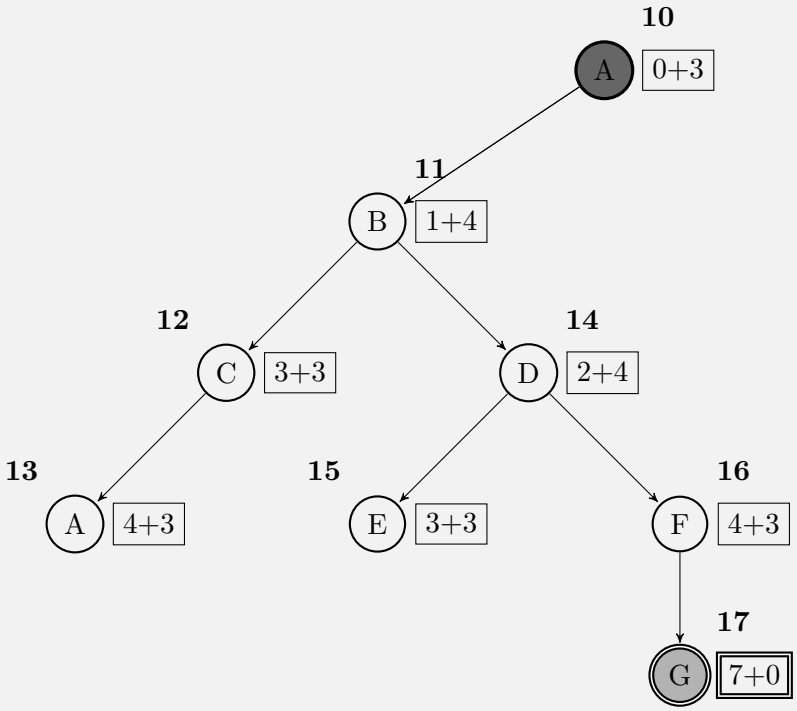
Profundidad 5



Profundidad 6

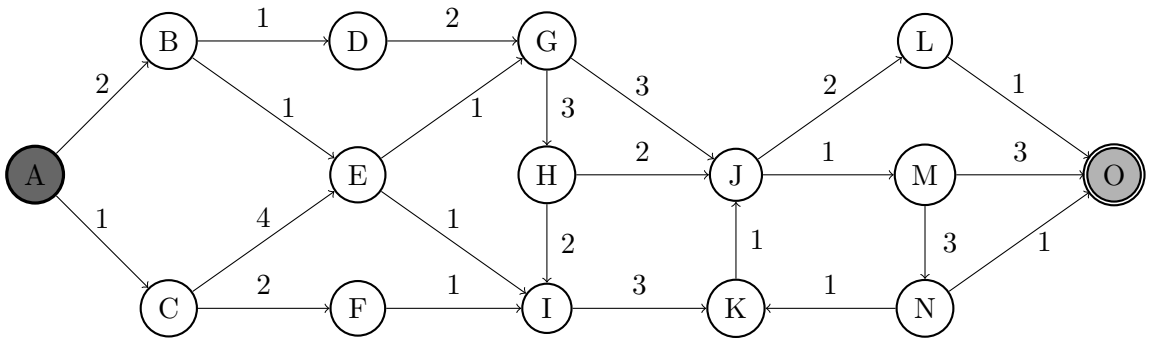


Profundidad 7



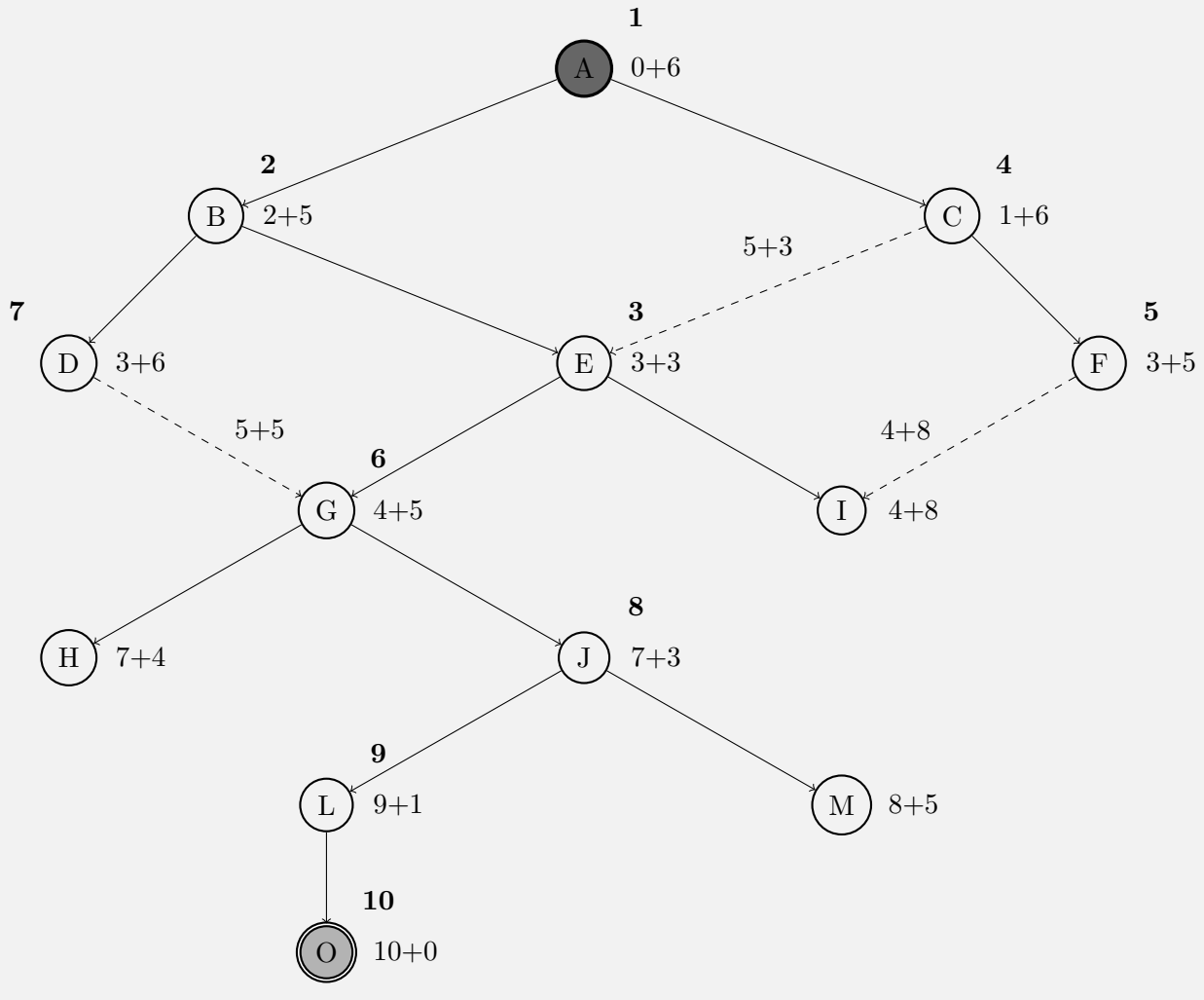
La función heurística es admisible. Se puede comprobar que para todos los valores de $h(n)$ que el coste real es siempre superior o igual.

16. Dado el siguiente grafo donde cada arco indica su coste y la tabla que indica la estimación del coste h hasta la solución, indica cual sería el árbol de búsqueda que se obtendría mediante el algoritmo de A* e IDA* para encontrar el camino entre el nodo A y el nodo O. Haz la generación de los nodos siguiendo el orden alfabético e indica claramente las reexpansiones de los nodos y los cambios de coste que aparezcan. ¿Es la función heurística admisible?

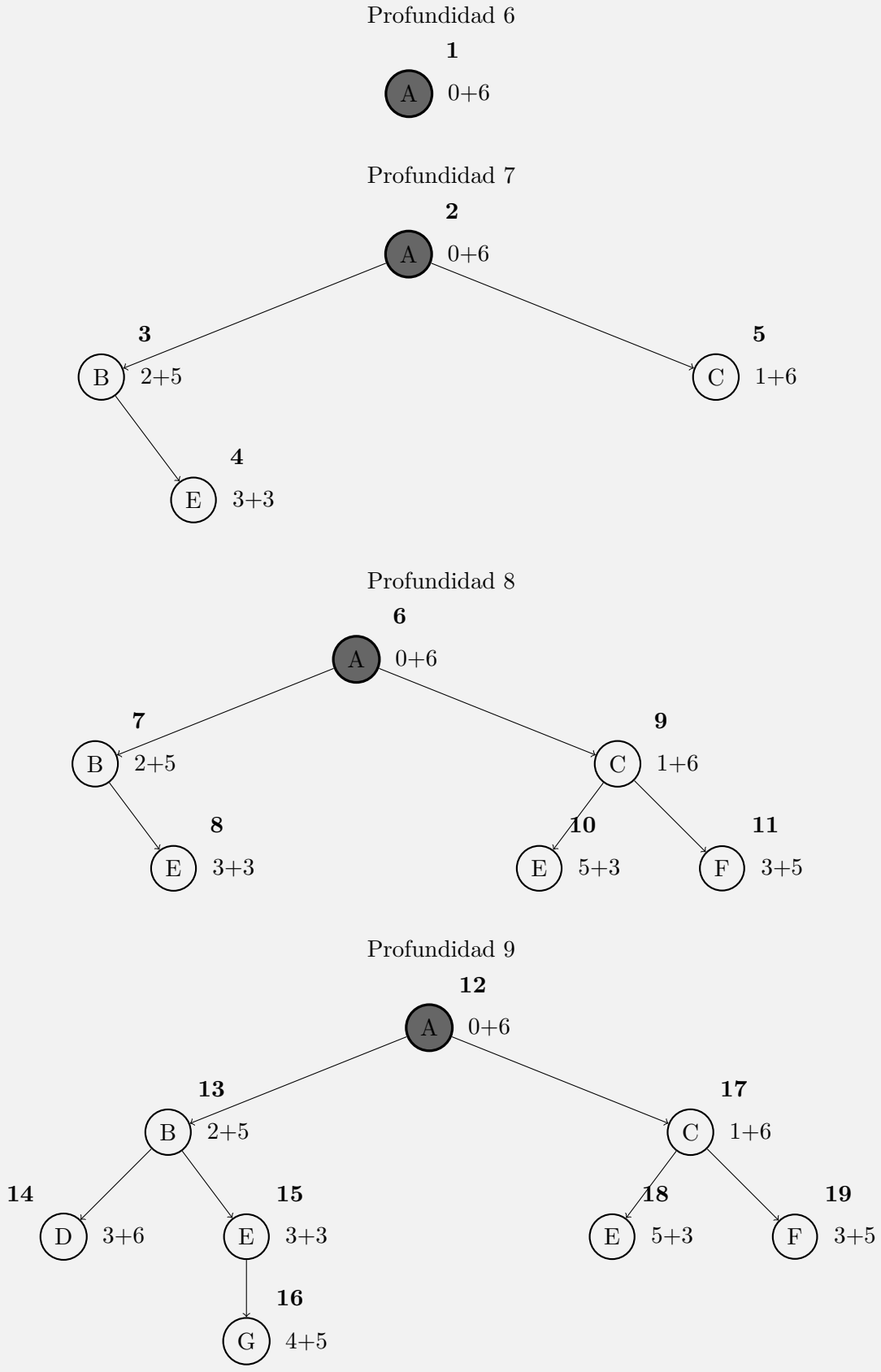


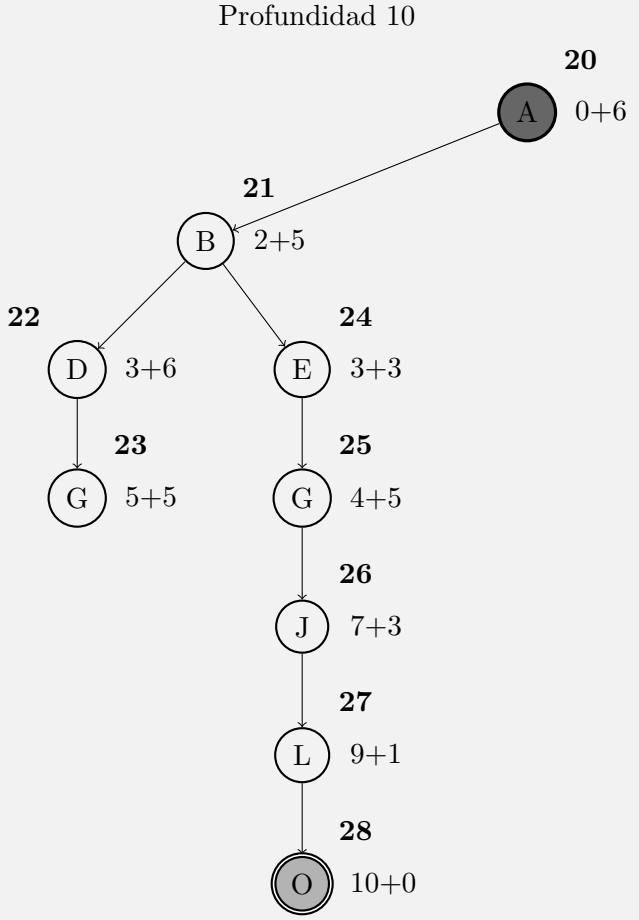
Nodo	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
h(nodo)	6	5	6	6	3	5	5	4	8	3	2	1	5	1	0

Árbol de búsqueda con A*



Árbol de búsqueda con IDA*





La función heurística no es admisible, se puede ver en el nodo M que tiene una h de 5 y un coste hasta la solución de 3.

2.1 Como plantear los problemas de búsqueda local

El objetivo de estos problemas es comentar las soluciones que se proponen a los diferentes elementos necesarios para resolver el problema de búsqueda que se plantea.

En un problema de búsqueda hay que decidir un conjunto de elementos:

- La representación del estado: Ha de ser válida y completa.
- La solución inicial: Ha de ser solución respecto a los criterios que plantea el problema, se debe evaluar el coste/posibilidad de obtenerla, si es necesario se deben poder obtener soluciones diferentes con el método propuesto, se debe evaluar su calidad.
- Operadores de búsqueda: Han de ser válidos, indicando unas condiciones de aplicabilidad y una función de transformación que genere soluciones, han de explorar correctamente el espacio de búsqueda, se ha de evaluar su factor de ramificación.
- Función heurística: Ha de incluir todos los objetivos del problema a optimizar, todos los objetivos han de ir en la sentido adecuado para optimizarlos, se ha de determinar si la ponderación de los diferentes objetivos es adecuada.

Al solucionar un problema mediante algoritmos genéticos hay que decidir tambien:

- La codificación del estado: Ha de ser válida y completa, han de poder representarse todos los estados.
- La población inicial: Se ha de poder generar mas de una solución inicial.
- Los operadores genéticos: Han de generar soluciones.

2.2 Problemas solucionados

6. Se han descubierto A fuentes de contaminación en un parque natural y se quieren colocar B (donde $B < A$) aparatos de descontaminación para mejorar la situación. Para ello se dispone de un mapa del parque que indica la posición de la estación de trenes donde se han almacenado todos los aparatos y de los A lugares donde se necesita colocar los aparatos de descontaminación. Además también se dispone del nivel de contaminación que hay alrededor de cada fuente, de un mapa de los desplazamientos (dirigidos) posibles de los aparatos en el territorio y del coste de cada desplazamiento. Cada aparato puede eliminar por completo la contaminación de una fuente, independientemente de su nivel.

   El objetivo es colocar los aparatos de manera que se minimice la contaminación total en el parque y el coste del recorrido (suma de desplazamientos) que harán los aparatos en el sentido "estación → fuente de contaminación".

   En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística). Hay que comentar la solución que se propone respecto a si es correcta, es eficiente, y es mejor o peor en comparación con otras alternativas. Y hay que justificar todas las respuestas.

   a) Se plantea solucionar el problema mediante Hill-Climbing, partiendo de una solución inicial sin ningún aparato y con un operador que coloca un aparato en una fuente de contaminación determinada, controlando que el número de los aparatos colocados sea como máximo B.

La solución vacía no debería considerarse solución por que que no cumple las condiciones, ya que una solución debe asignar B aparatos.

No obstante, podriamos obviar este problema dado el operador que tenemos. A cada paso se coloca un aparato en una fuente de contaminación, la única restricción es no superar el número de asignaciones máximo, esto nos garantiza que la exploración podrá pasar del espacio de no soluciones al de soluciones.

Si el heurístico es bueno, partiendo de la solución vacía, este sería un buen operador ya que a cada paso del HC se escogerá la estación que más convenga hasta que se hayan puesto los B aparatos o ya no merezca la pena poner más (porque lo que se minimiza en contaminación no lo vale con el coste de desplazamiento de más que tenemos que hacer). El factor de ramificación del operador será el número de fuentes de contaminación por el número de aparatos disponibles ($O(A \cdot B)$).

b) Se plantea solucionarlo mediante Hill-Climbing, partiendo de una solución inicial con B aparatos colocados aleatoriamente, y utilizando como función heurística la suma de los costes de desplazamiento de la estación a cada una de las B fuentes.

La solución es correcta puesto que respeta la restricción de que debe haber de B aparatos. También es barata, pero, al ser al azar, no tenemos ninguna garantía sobre su calidad.

La función heurística no es correcta, ya que solo tiene en cuenta el coste del desplazamiento y no tiene en cuenta la contaminación.

c) Se plantea solucionarlo mediante Hill-Climbing, partiendo de una solución inicial con B aparatos colocados aleatoriamente, y utilizando como función heurística la suma de los costes mínimos de los recorridos "estación → fuente de contaminación" multiplicada por la suma de los niveles de contaminación en correspondientes a los B aparatos.

Solución inicial, mismos comentarios que en el apartado anterior.

La función heurística no es correcta, ya que queremos minimizar el primer multiplicando mientras que queremos maximizar el segundo. Tal como esta la funcion, si la minimizaramos colocariamos los aparatos en los puntos de menor contaminación. Una posible solución sería restar la contaminación al recorrido, pero la diferencia de unidades entre los dos factores podría asignar más peso a un factor que al otro. Otra solución sería pasar el segundo mutiplicando a dividir, de esta forma, cuanto más grande sea el denominador, más pequeño será el resultado.

d) Se plantea solucionarlo mediante Hill-Climbing, partiendo de una solución inicial alcanzada colocando los B aparatos ordenadamente según el coste mínimo "estación → fuente de contaminación" y empezando con el que tiene coste menor. Se plantea como operador mover un aparato a cualquier fuente cuyo producto de "coste mínimo estación → fuente" por "nivel de contaminación" sea menor que el actual.

Esta solución inicial es mas costosa que las anteriores, ya que requiere ordenar las fuentes, concretamente el coste es $O(A \cdot log(A))$. Dado que el coste de desplazamiento no es el único criterio a optimizar utilizar solo el coste de desplazamiento no nos garantiza nada sobre la bondad de la solución inicial respecto al objetivo.

El operador es correcto respecto a respetar la restricción de colocar B aparatos. La solución inicial parte de B aparatos colocados y solo cambiamos aparatos de sitio, no ponemos ni quitamos ninguno. En este caso el factor de ramificación sería $O(A \cdot B)$. No obstante, el operador estaría optimizando la suma de los productos para cada fuente de contaminación de su distancia y nivel de contaminación, ya que prefiere movimientos en los que el producto de estos dos valores es menor, esto minimizaría los recorridos, pero también escogería los puntos de menos contaminación, que es precisamente lo contrario de lo que queremos.

e) Se plantea solucionarlo mediante algoritmos genéticos: se usan individuos de A bits y como población inicial se generan n individuos donde en cada uno hay exactamente B bits a 1. La función de idoneidad es la suma de los costes mínimos "estación → fuente de contaminación" más la contaminación total residual del parque multiplicada por una constante. Como operadores se usan los habituales de cruce y mutación.

La población inicial es correcta ya que cada individuo representa una solución diferente. El coste de hallar la solución inicial es $O(n)$, siendo n el número de individuos.

La función de idoneidad es correcta ya que queremos minimizar el coste por desplazamiento y también la contaminación del parque, en este caso usamos la contaminación que no se limpia. El hecho de multiplicar la contaminación por una constante dará más peso a la contaminación, que ya parece lo adecuado.

El operador Cruce no es correcto, ya que podría pasar que al cruzar dos individuos, el individuo resultante tuviera más o menos de B bits a 1, y eso sería una solución no válida.

El operador Mutación tampoco es correcto ya que siempre nos dará soluciones con un bit mas o un bit menos que B.

9. Tenemos una pequeña flota de C camiones que utilizamos para repartir mercancías y cada día tenemos que determinar qué ruta ha de seguir cada camión para abastecer un conjunto de ciudades por todo el país. El objetivo es que todos los camiones acaben la jornada aproximadamente a la misma hora, por lo que el número de kilometros que ha de recorrer cada camión ha de ser muy parecido, y que recorran en total el mínimo número de kilómetros.

   Disponemos de un mapa de carreteras que nos dice la distancia en kilómetros entre cada ciudad donde hemos de dejar nuestra mercancía, suponemos que todos los camiones parten de la misma ciudad y han de volver a ella al final del día, cargan al principio de la jornada todo lo que han de repartir, han de pasar sólo una vez por cada ciudad.

   Una posible solución a éste problema se puede obtener mediante el uso de algoritmos de búsqueda local. En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística,...). Comenta muy brevemente la solución que se propone respecto a si es correcta y si es mejor/peor respecto a otras alternativas posibles. Justifica tus respuestas.

   a) Usar Hill Climbing. Como solución inicial asignamos al azar a cada camión un número aproximadamente igual de ciudades, recorriéndolas en orden también al azar. Como operadores usamos el intercambiar dos ciudades entre los recorridos de dos camiones e intercambiar las posiciones de dos ciudades en el recorrido de un camión. La función heurística es la siguiente:

$$h'(n) = \sum_{i=1}^{C} \left(\frac{LR_i}{\sum_{j=1}^{C} LR_j} - \frac{1}{C} \right)$$

   donde LR_i es la longitud del recorrido del camión i.

Por lo general, la aleatoriedad nos genera una solución inicial de forma rápida, aunque la bondad de esta solución será variable en cada ejecución. El balanceo de carga de los recorridos, respecto del número de ciudades por recorrido se cumple, pero eso no implica necesariamente que se cumpla por longitud de los recorridos, que sería lo ideal. Por lo tanto, es importante tener un buen heurístico que nos permita mejorar este aspecto. El heurístico está calculando lo siguiente:

$$h'(n) = \sum_{i=1}^{C}(\frac{LR_i}{\sum_{j=1}^{C} LR_j} - \frac{1}{C}) = [\sum_{i=1}^{C}(\frac{LR_i}{\sum_{j=1}^{C} LR_j})] - 1 = \frac{\sum_{i=1}^{C} LR_i}{\sum_{j=1}^{C} LR_j} - 1 = 1 - 1 = 0.$$

Este heurístico no está dándonos ninguna información, pese a lo que inicialmente pareciera. Dado que la bondad siempre es cero, ninguna solución sucesora de la inicial podría mejorarlo y por solución final se obtendría la inicial. De esta forma, el resultado es obtenido aleatoriamente.

Con respecto a los operadores de transformación hay que tener en cuenta que *2.* siempre mantendrá los recorridos balanceados (salvo sobre el que se aplique), que es lo deseado por lo general. *1.* mantiene el balanceo en número de ciudades por recorrido, pero no necesariamente en longitud (sobre los dos recorridos implicados).

Factores de ramificación:

1) Si tenemos un recorrido de un camión C_i y otro C_j con longitud del recorrido L_{C_j}, en número de ciudades, el factor de ramificación de *1.* con origen en C_i y destino C_j es de $FR_{Op_1} = N - L_{C_j}$. Para toda ciudad es, entonces, del orden de $O(N^2)$.

2) Si L_{C_i} es la longitud, en cada momento, del recorrido del camión C_i, en número de ciudades, el factor de ramificación es de $FR_{Op_2} = L_{C_i} - 1$. El rango es $0 \leq FR_{Op_2} \leq N - 1$. Para toda ciudad es, entonces, del orden de $O(N^2)$.

El problema que se nos plantea con estos dos operadores es la localidad. Dada una solución inicial, las soluciones derivadas de esta son las derivadas de los posibles intercambios. Esto nos mantiene, siempre, la longitud de los recorridos en número de ciudades, como ya se ha dicho, lo que supone que el resto del espacio de estados queda fuera del potencial de las operaciones e inexplorado.

b) Usar Hill Climbing. Como solución inicial asignamos todas las ciudades a un camión, estableciendo el recorrido inicial mediante una estrategia avariciosa de manera que intentemos minimizarlo. Como operadores usamos el mover una ciudad del recorrido de un camión a otro e intercambiar las posiciones de dos ciudades en el recorrido de un camión. La función heurística es la siguiente:

$$h'(n) = \prod_{i=1}^{C} LR_i$$

El uso de una técnica avariciosa supondrá un coste cuadrático para hallar el primer recorrido, eligiendo, a cada paso, el menor de los valores posibles para las ciudades sin asignar. Al estar todas las ciudades asignadas a un mismo camión, la solución está muy desbalanceada (en todos los aspectos), y eso no se desea. Para encontrar una solución buena debemos alejarnos mucho en el espacio de estados de esta solución inicial, por lo que tendrá que ser destruida completamente, haciendo que el coste de hallarla se puerda completamente.

Para modificar este estado y pasar a uno más balanceado, deberemos usar un heurístico que nos guíe en ese camino, pero inicialmente la solución inicial será, por lo general, bastante mala.

El heurístico está calculando lo siguiente:

$$h'(n) = \textstyle\prod_{i=1}^{C} LR_i$$

Y eso quiere decir que si algún $LR_i = 0 \rightarrow h'(n) = 0$. Y como en la solución inicial todos los recorridos son vacíos salvo uno, eso quiere decir que el heurístico será cero, y ninguna otra solución sucesora puede mejorar ese valor por lo que la solución inicial será la final.

Con toda seguridad la solución de este método será peor que la encontrada por el apartado *a)*, ya que en el apartado *a)* la aleatoriedad da cierta posibilidad a obtener mejores resultados.

Con respecto a los operadores de transformación hay que decir que la aplicación sucesiva del operador *1.* podría desbalancear una solución. Pese a esto, este operador puede resultar útil para contrarrestar un efecto de desbalanceo, y como la solución inicial está muy desbalanceada este operador puede ser muy útil. Con el operador *2.*, que realiza un intercambio sobre el mismo recorrido, no se altera el balanceo del resto.

Factores de ramificación:

1) Si tenemos tenemos C recorridos de camiones y un recorrido de un camión C_i, el factor de ramificación de *1.* con origen en C_i y destino cualquier C_j, $i \neq j$, es de $FR_{Op_1} = C - 1$. Así que para toda ciudad sería del orden de $O(N * C)$.

2) Si L_{C_i} es la longitud, en cada momento, del recorrido del camión C_i, en número de ciudades, el factor de ramificación es de $FR_{Op_2} = L_{C_i} - 1$. El rango es $0 \leq FR_{Op_2} \leq N - 1$. Para toda ciudad es, entonces, del orden de $O(N^2)$.

En esta selección de operadores no aparece el problema de la localidad ya que el operador *1.* varía las longitudes en número de ciudades por recorrido lo que permite explorar todas las posibles combinaciones de ciudades a recorridos. Cabe destacar que el operador *2.* por su parte sigue teniendo el mismo problema.

c) Usar Hill climbing. Como solución inicial escogemos las C ciudades más cercanas a la ciudad origen como la primera ciudad a visitar por cada camión, como segunda ciudad en el recorrido de cada camión escogemos la más cercana a la primera que no esté ya asignada, y así sucesivamente hasta asignar todas las ciudades. Como operadores usamos el mover una ciudad del recorrido de un camión a otro, intercambiar las posiciones de dos ciudades en el recorrido de un camión e intercambiar dos ciudades entre los recorridos de dos camiones. La función heurística es la siguiente:

$$h'(n) = \frac{C \cdot (C-1)}{2} - \sum_{i=1}^{C} \sum_{j=i+1}^{C} \frac{LR_i}{LR_j}$$

La solución inicial estará balanceada en número de ciudades, no esta claro que la forma de generar el recorrido balancee las distancias.

Aplicar este sistema para crear la solución inicial es identico en el caso peor al anterior, la primera asignacion requiere escoger las C ciudades más cercanas a la ciudad origen ($O(nlog(N))$ y luego hay que escoger para cada ciudad que asignemos a cada recorrido su ciudad más cercana ($O(N^2)$)

En el heurístico el primer elemento de la resta es un factor constante por lo que no tiene ningún efecto aparte de establecer el rango. El segundo elemento debería acercarse al valor del primero cuanto más equilibrados estén los recorridos en longitud, que es lo que realmente se desea. Por lo tanto, este heurístico dará cero cuando esté totalmente equilibrada la solución.

El problema es que, la solución obtenida, pese a estar equilibrada no tiene porque ser de longitud mínima, lo que hace el heurístico es intentar equilibrar la proporción de las longitudes de cada recorrido.

Los tres operadores son correctos y generan soluciones, permitiendo generar todo el espacio de búsqueda. El operador intercambiar equivale a dos operaciones de mover, pero puede ser necesario ya que es posible que no se puedan aplicar las dos operaciones separadamente y si como una operacion de intercambio.

Factores de ramificación:

1) Si tenemos tenemos C recorridos de camiones y un recorrido de un camión C_i, el factor de ramificación de *1.* con origen en C_i y destino cualquier C_j, $i \neq j$, es de $FR_{Op_1} = C - 1$. Así que para toda ciudad sería del orden de $O(N * C)$.

2) Si tenemos un recorrido de un camión camión C_i y otro C_j con longitud del recorrido L_{C_j}, en número de ciudades, el factor de ramificación de *1.* con origen en C_i y destino C_j es de $FR_{Op_1} = N - L_{C_j}$. Para toda ciudad es, entonces, del orden de $O(N^2)$.

3) Si L_{C_i} es la longitud, en cada momento, del recorrido del camión C_i, en número de ciudades, el factor de ramificación es de $FR_{Op_2} = L_{C_i} - 1$. El rango es $0 \leq FR_{Op_2} \leq N - 1$. Para toda ciudad es, entonces, del orden de $O(N^2)$.

En esta selección de operadores no aparece el problema de la localidad ya que el operador *1.* varía las longitudes en número de ciudades por recorrido lo que permite explorar todas las posibles combinaciones de ciudades a recorridos. Cabe destacar que el operador *2. y 3.* por su parte siguen teniendo el mismo problema.

d) Usar algoritmos genéticos. Donde cada ciudad está representada por tantos bits como sean necesarios para codificar el valor C, la tira de bits contiene concatenados los bits de todas las ciudades, es decir, representamos en la tira de bits el número del camión que ha de recorrerla. Como operadores utilizamos los operadores habituales de cruce y mutación.

La codificación escogida no representa el orden en que un camión va a recorrer las ciudades que se le han asignado. En este problema usar una codificación binaria es difícil si se quiere poder representar el orden de recorrido de las ciudades ya que cada recorrido tendrá una longitud variable.

Además matener la coherencia de la representación al aplicar los operadores genéticos sería bastante difícil.

10. La sala de cine "Lo nunca visto" desea realizar una planificación estratégica de las películas a proyectar durante un año (52 semanas) siguiendo varios criterios comerciales. Cada película tiene asociada varias

informaciones relevantes para este problema: tipo (infantil/adulto), previsión de beneficio semanal, índice de calidad. La programación anual debe contener al menos un 15 % de películas infantiles y un 40 % de películas para adultos. Se desea maximizar el beneficio anual de proyección de películas, pero sin dejar de lado la calidad. Por este motivo, el índice global de calidad no debe ser inferior a una cota predeterminada (Q). En el caso de que una película deba proyectarse más de una semana, dichas semanas deberán ser consecutivas. Las estrategias comerciales exigen que una película no se proyecte más de 8 semanas.

Teniendo en cuenta el escenario descrito, comenta brevemente las diversas propuestas que se describen en los apartados siguientes. Valora si son correctas o no, si son eficientes o no, si son mejores o peores que otras alternativas. Justifica tus respuestas.

a) Usar un Hill-climbing. Como estado inicial asignamos aleatoriamente una película distinta a cada una de las 52 semanas. Como operador usamos Asignar-película (título, num-semana), que sustituye la película asignada una semana por otra nueva o por alguna de las ya asignadas a otras semanas.

La forma de generar la solucion inicial no nos da ninguna garantia de que cumpla las condiciones impuestas (% de peliculas infantiles y adultos), por lo tanto no genera soluciones.

El operador asignar pelicula no tiene en cuenta las condiciones que se imponen, si la pelicula ya esta asignada solo se puede asignar a alguna de las semanas contiguas para respetar las condiciones del problema. Tampoco podemos romper una secuencia de peliculas contiguas, solo se puede quitar la primera o a ultima. Tampoco podemos poner peliculas mas de 8 semanas.

Si añadimos estas condiciones podriamos explorar todas las posibles configuraciones validas y el factor de ramificacion en el caso peor seria O(52 * num peliculas)

b) Usar un Hill-climbing. Como estado inicial asignamos dos películas infantiles (4 semanas cada una) y seis películas para adultos (4 semanas cada una), el resto de las semanas se asigna aleatoriamente. Como operador usamos el mismo del apartado a.

En este caso la solucion inicial cumpliria los porcentajes de peliculas que se piden, pero el enunciado no dice exactamente como se hace la asignacion aleatoria, esta debería colocar una pelicula distinta el resto de semanas para evitar violar las condiciones del problema.

El coste de hallar esta solucion es barato, pero no podemos decir nada sobre la calidad de la solucion.

c) Usar un Hill-climbing. Partimos del mismo estado inicial del apartado b con el control adicional de que el índice de calidad no sea inferior a Q. Al operador de asignación le añadimos como condiciones de aplicabilidad que mantenga las proporciones mínimas de películas infantiles y para adultos, que ninguna película supere las 8 semanas y que se respete la cota inferior Q. Como función heurística usaremos h1'(n): la suma de la previsión de beneficio semanal para cada película proyectada multiplicada por el número de semanas asignado.

Si intentamos forzar que la solucion inicial tenga un indice de calidad superior a Q aumentamos el coste de hallar la solucion. Estaremos obligados a ordenar las peliculas y escogerlas para superar ese valor. Tampoco podemos decir nada sobre la calidad global de la solucion ya que involucra el beneficio y no se tiene en cuenta en la creacion de la solucion inicial.

El operador de busqueda añade casi todas las condiciones que mantienen dentro del espacio de busqueda salvo la contigüidad de las semanas de las peliculas iguales. El factor de ramificacion en el caso peor sera el mismo que antes, pero evidentemente con estas restricciones las elecciones reales se habran reducido considerablemente.

La funcion heuristica incluye solamente el beneficio, pero dado que sobre la calidad se tiene que cumplir solamente que no se baje del valor Q las soluciones que encontrariamos serian adecuadas respecto a lo que se pide.

De hecho el valor de calidad no es un factor que se haya de optimizar, si lo incluyeramos en la funcion heuristica nos podriamos encontrar con cosas diferentes de lo que esperamos dependiendo de como lo

combinaramos con el beneficio. Soluciones con mayor beneficio y peor calidad podrian parecer mejores que soluciones con menor beneficio pero mayor calidad.

d) Usar un Hill-climbing. Como heurístico usaremos h2'(n): h1'(n) + suma del índice de calidad de cada película proyectada multiplicado por el número de semanas asignado.

Este apartado demuestra lo que se comentaba sobre el papel que juega la calidad en el apartado anterior. De hecho esta restriccion nos indica que forma parte del espacio de soluciones y que no.

El enunciado no indica que hayamos de maximizar la calidad, solo impone que posibles soluciones no queremos considerar (las que tengan un valor inferior a Q) por lo tanto todas las que superen ese valor son iguales para nosotros.

Si quisieramos tambien maximizar la calidad deberiamos decidir cual es su relacion respecto al beneficio.

e) Usar algoritmos genéticos. Cada individuo se representa mediante 52 tiras de bits. La longitud de la tira de bits es suficiente para codificar el número de identificación de todas las películas. Como población inicial se generan n individuos con los criterios de estado inicial descritos en b y c. Como función de fitness usamos h3'(n): h1'(n) + (suma del índice de calidad de cada película proyectada multiplicado por el número de semanas asignado - Q). Como operadores usamos el operador de cruce habitual, pero trabajando siempre con tiras completas de bits (la identificación de una película), es decir, un punto de cruce nunca parte el identificador de una película.

La representacion es adecuada, representa las 52 semanas y la asignacion de la pelicula que le corresponde.

Se generan n individuos siguiendo los criterios de los apartados b y c, aunque se deberian especificar mejor para poder garantizar cierta diversidad entre las soluciones.

La funcion heuristica tiene el problema del apartado d, lo unico que hacemos es tener el origen del valor a 0 en lugar de en Q.

Utilizar el operador de cruce solamente y no permitir cruces dentro de los identificadores de pelicula hace que no aparezcan peliculas nuevas de las que ya hay en laas soluciones iniciales. Ademas no tenemos garantias de que las soluciones resultantes cumplan las condiciones del problema.

16. Tenemos que planificar la topología de interconexión de un conjunto de routers que están distribuidos por el campus nord, de manera que podamos canalizar todo su tráfico hacia el equipo de comunicaciones que lo reenvía a internet. Para cada router sabemos su localización en el campus (supondremos que tenemos un sistema de coordenadas en el que podemos situar cada router) y el ancho de banda del tráfico directo que debe distribuir (el que no le llega de otros routers). También conocemos las coordenadas del equipo de comunicaciones. Cada router puede estar conectado a otro router o al equipo de comunicaciones exterior directamente (la topología ha de ser un árbol). Disponemos de tres tipos de router (tipos A, B y C) capaces de distribuir hasta cierto ancho de banda máximo cada uno. Para distribuir el tráfico directo hay suficiente con un router de tipo A. El coste de cada router es proporcional al tipo (el tipo B cuesta el doble que el A y el C el doble que el B). Para conectar cada router necesitamos instalar un cable de cierta longitud (supondremos que es la distancia euclídea entre las coordenadas de los dos equipos conectados) el coste de este cable es proporcional a la longitud y se multiplica por el coste del tipo del router que envía por ese cable. El coste del equipo de comunicaciones exterior es fijo y no lo tendremos en cuenta.

    El objetivo es decidir el tipo de los routers dependiendo del ancho de banda que deben soportar y la forma de interconectarlos, de manera que se que minimice el coste de la instalación.

    En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística). Hay que comentar la solución que se propone respecto a si es correcta, es eficiente, y es mejor o peor en comparación con otras alternativas. Y hay que justificar todas las respuestas.

    a) Usar Hill climbing. Como solución inicial conectamos directamente todos los routers al equipo de comunicaciones exterior. Como operadores usamos conectar un router a otro y desconectar un router de otro. La función heurística es la suma de los costes de los routers.

Estado inicial es correcto siempre que supongamos que los routers son de tipo A, por ejemplo, sino no seria una solución correcta por incompleta.

Los **Operadores** no son correctos porque al aplicar cualquiera de ellos se obtiene un estado no solución. Al aplicar el operador de *conectar un router con otro* no se tiene en cuenta la topologia de la solución. En el siguiente operador, desconectar un router de otro partiendo de un estado solución, quedaría un router o un conjunto de routers que no tendría conexión con el equipo de comunicación exterior. Además, no se puede explorar todo el espacio de soluciones al no poder cambiar el tipo de router.

La **función heurística** es incorrecta al no tener en cuenta el coste del cableado.

b) Usar Hill climbing. Como solución inicial calculamos el árbol de expansión mínima de los routers y el equipo exterior y asignamos a cada router el tipo C. Como operador usamos cambiar la conexión de un router o equipo exterior a otro. La función heurística es la suma de los costes del cableado.

Estado inicial por lo general no es correcto. Por definición el resultado de calcular el árbol mínimo de expansión es un árbol, por lo tanto cumple la topología del problema. No lo indica pero podemos suponer que el criterio es la distancia entre los diferentes routers. La complejidad del algoritmo es $O(m \log m)$ siendo m el número de aristas del grafo completo. Seria solución si se garantizará que cumple la condición del ancho de banda.

Operadores es correcto pero incompleto al no poderse cambiar el tipo de router, por tanto si el estado inicial fuera solución ya sería la mejor al ser un árbol de expansión mínimo. Además no se garantiza en ningún momento la restricción del ancho de banda. También comentar que el **factor de ramificación** es $O((r-1)^2 + r * 2)$, siendo r el número de routers.

Función heurística es incorrecta al no considerar el coste del router. Un alternativa sería añadir a todas las distancias una unidad, así tendríamos en cuenta el precio del router.

c) Usar Hill climbing. Como solución inicial conectamos cada router con el router más cercano y conectamos aleatoriamente uno de los routers al equipo exterior, asignamos a cada router el tipo A. Como operador usamos cambiar la conexión de un router a otro router o equipo exterior por otra distinta si no superamos su capacidad y cambiar el tipo del router. La función heurística es la suma de los costes del cableado más el coste de los routers.

Estado inicial nos encontramos en la misma situación que en el caso anterior. Para poder garantizar que es solución tendríamos que verificar la restricción del ancho de banda. Por el resto seria una solución muy parecida al anterior, con la diferencia que en esta ocasión se asignan routers A en vez de C. En esta ocasión tampoco se tiene en cuenta el ancho de banda.

Operadores los operadores nos permiterian movernos por todo los estados, por tanto serian correctos. De todas formas tendría que existir una combinación de los dos ya que un router más barato nunca seria substituido por otro más caro pero si podría existir la combinación de los dos operadores que obtubiera un mejor resultado. El factor de ramificación es muy parecido al anterior apartado. **Función heurística** incluye correctamente todos los elementos que queremos optimizar.

d) Usamos algoritmos genéticos. Supondremos que los routers y el equipo exterior están numerados consecutivamente, supondremos que para codificar ese número hacen falta n bits, la codificación es una tira $r \cdot n$ de bits, donde r es el número de routers, cada grupo de n bits corresponden a un router siguiendo el orden de la numeración. Como operadores usamos los operadores de cruce y mutación habituales. La función heurística la suma de los costes de cada cable dividido por el coste del router que envía por él.

Codificacion es correcta en lo que respecta a la representación de las interconexiones, cada grupo de bits es el router al que se conecta, pero no se representa el tipo de los routers.

Operadores no son correctos porque no tienen en cuenta las restricciones del problema. Un cruce o una mutación deberían controlar que lo que obtenga es un grafo y que no se supere la capacidad de los routers.

Función heurística es equivalente a la suma de las distancias del cable, por lo que es incompleta.

18. El canal musical de televisión clásica *XL Recordings* desea realizar una planificación del contenido a transmitir durante un día. Se ha decidido que durante un día se pueden emitir 200 vídeos y quieren poder determinar qué vídeos han de emitirse y en qué orden. La cadena dispone de una videoteca con V vídeos de entre los que elegir.

    Para cada vídeo conocemos los ingresos por publicidad que genera su emisión y la popularidad del grupo musical del vídeo (valor de 1 a 5).

    El objetivo de la cadena es maximizar los ingresos por publicidad, evitando que el número de vídeos con cierto valor de popularidad sea superior al 40 %. Para no aburrir a la gente, no debemos emitir ningún vídeo mas de una vez, tampoco debemos poner demasiado juntos los vídeos de la misma popularidad.

    En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística). Hay que comentar la solución que se propone respecto a si es correcta, es eficiente, y es mejor o peor en comparación con otras alternativas. Y hay que justificar todas las respuestas.

    a) Usar Hill climbing. Como solución inicial elegimos 40 vídeos de cada popularidad y los colocamos aleatoriamente en la secuencia de emisión. Como operador intercambiar el orden de emisión de dos vídeos. Como función heurística usamos

    $$h'(n) = \sum_{i=1}^{200} IngresosPublicidad_i$$

    Donde $IngresosPublicidad_i$ es la cantidad que se ingresa por publicidad con el vídeo i.

La solución inicial es válida, ya que se evita tener más del 40 % de los vídeos de una misma popularidad (se tiene exactamente el 20 % de cada tipo). Es también una solución de coste bajo. Pero no es una buena solución, ya que la generamos sin usar información respecto al objetivo del problema.
Sería una solución mejor alguna en la que por cada popularidad se escogiesen los 40 vídeos que generan más ingresos por publicidad, esto reduciría probablemente el número de iteraciones del Hill climbing.
El operador propuesto (intercambio de dos vídeos) conlleva que todas las soluciones a las que se podrá llegar son ésas cuyos vídeos pertenecen a la solución inicial escogida. Su factor de ramificación seria $O(200^2)$. Esto implica que no es posible explorar todo el espacio de soluciones, creando una restricción claramente excesiva, puesto que es posible que existan soluciones mucho mejores que incluyan otros vídeos. Tampoco comprueba que haya demasiados videos de la misma popularidad juntos.
Aunque el heurístico tiene en cuenta nuestro objetivo de maximizar los ingresos por publicidad, no se tiene en cuenta el hecho de que se buscan soluciones en las que vídeos con la misma popularidad no estén demasiado juntos. Es, por lo tanto, un heurístico incompleto.

    b) Usar Hill climbing. Como solución inicial elegimos los 200 vídeos que generan mayores ingresos por publicidad. Como operadores usamos cambiar un vídeo de la solución por otro que no este emitido e intercambiar el orden de emisión de dos vídeos.

Partimos de una solución inicial que gran probilidad no cumple las restricciones para ser solución, ya que no se tiene en cuenta la restricción de que el número de vídeos con cierta popularidad no puede ser mayor que el 40 %.
Los operadores tampoco comprueban que las soluciones generadas cumplan las restricciones del problema, deberían hacerlo para poder asegurar el hallar una solución. Aperte de esto con estas operaciones podríamos generar todos los estados posibles.
El factor de ramificación del operador de intercambio de dos vídeos es $O(200^2)$ y la ramificación de cambiar por un vídeo no emitido $O(videos * 200)$.

    c) Usar Hill climbing. Usamos como operador cambiar un vídeo de la solución por otro que no este emitido, como función heurística:

$$h'(n) = \sum_{i=1}^{200} f(IngresosPublicidad_i)$$

donde $f(IngresosPublicidad_i)$ vale 0 si el porcentaje de vídeos de la misma popularidad que el vídeo i supera el 40 % o $IngresosPublicidad_i$ en caso contrario.

Como en el caso anterior, nos encontramos que los operadores no aseguran que las soluciones a las que accedemos son válidas.

Dada la función heurística propuesta, en este caso si usamos Hill climbing y partimos de una solución válida, nunca escogeríamos a una solución no válida durante la búsqueda, ya que si partimos de una solución válida, e intercambiamos un vídeo por otro que infringe la restricción del 40 %, al evaluar la función heurística nos encontramos que el valor de la nueva solución es peor.

En este caso el heurístico tiene en cuenta una de las restricciones del problema en lugar de controlarla en los operadores y optimiza los ingresos por publicidad. No obstante, el heurístico todavía sería incompleto ya que no se tiene en ningún momento en cuenta la cercanía de los vídeos de la misma popularidad.

d) Usar algoritmos genéticos. Supondremos que hacen falta b bits para codificar el identificador de un vídeo. Representamos cada individuo como una tira de $200 \cdot b$ bits. Como población inicial, generar n individuos escogiendo 200 vídeos aleatoriamente. Como operadores utilizamos solamente el operador de cruce.

La representación del individuo es correcta ya que podemos codificar cualquier solución (conjunto de 200 videos).

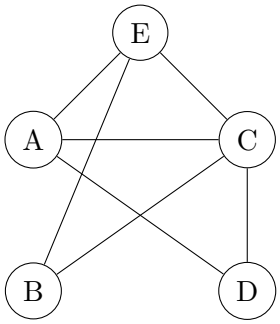
La forma de generar la solución inicial nos asegura tener n individuos distintos, lo cual es necesario para este tipo de algoritmos.

El operador de cruce no nos asegura obtener soluciones válidad ya que no comprueba ninguna de las restricciones del problema y ademas puede generar soluciones con videos repetidos. El no utilizar el operacion mutación tampoco es bueno porque el operador de cruce tampoco nos asegura que vayamos a poder generar todos los identificadores de video existentes.

3.1 Problemas solucionados

19. Dado el siguiente grafo de restricciones donde cada restricción es una condición de desigualdad y los siguientes dominios para las variables:

A={1,2}
B={2,3}
C={1,2}
D={1,2,3}
E={1,2,3}



Haz la ejecución del forward checking hasta encontrar la primera solución

```
1) Asignación A= 1

Variable A restringe variable C
C={2 }
Variable A restringe variable D
D={2 3 }
Variable A restringe variable E
E={2 3 }

2) Asignación B= 2

Variable B restringe variable C

C={}

3) Asignación B= 3

Variable B restringe variable E
E={2 }
```

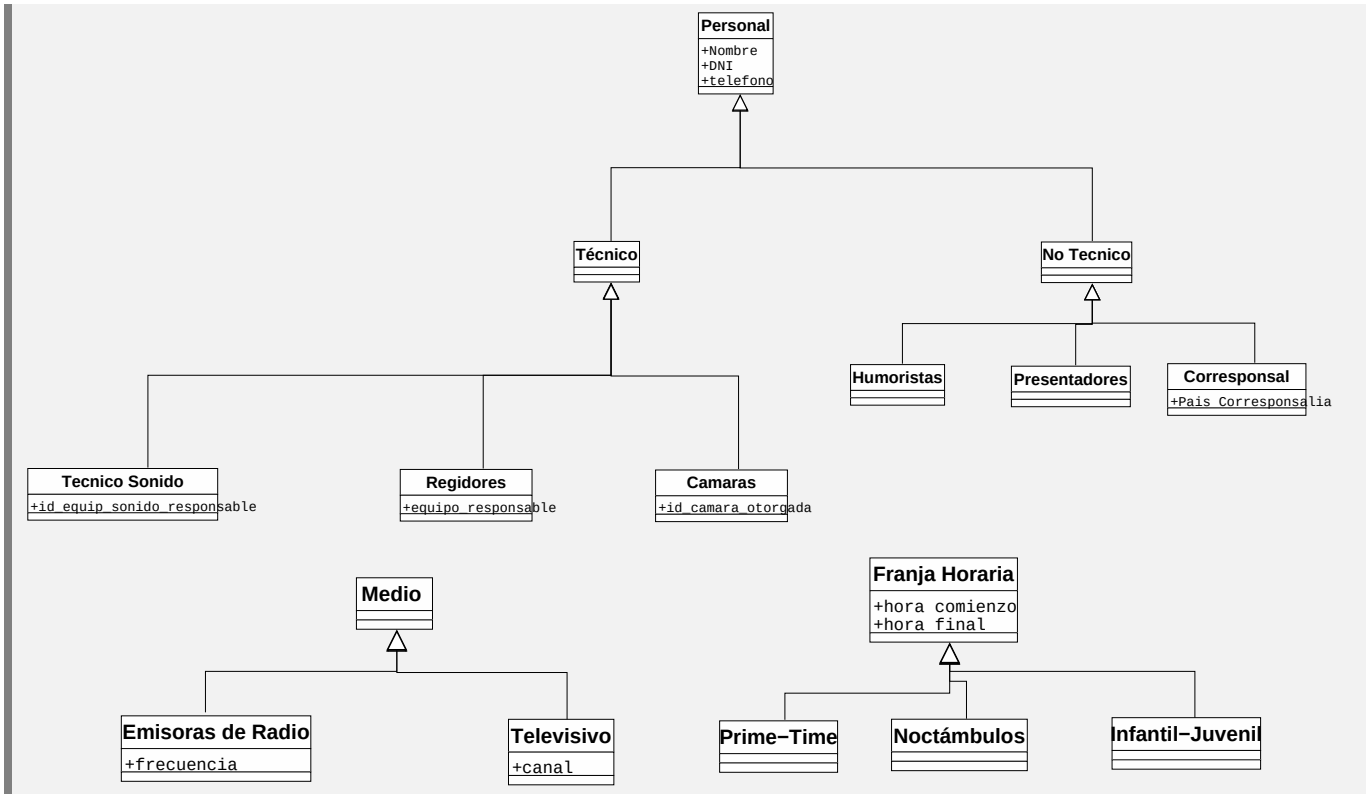
```
4) Asignación C= 2

Variable C restringe variable D
D={3 }
Variable C restringe variable E
E={}

Backtracking a A

5) Asignación A= 2

Variable A restringe variable C
C={1 }
Variable A restringe variable D
D={1 3 }
Variable A restringe variable E
E={1 3 }

6) Asignación B= 2

7) Asignación C= 1

Variable C restringe variable D
D={3 }
Variable C restringe variable E
E={3 }

8) Asignación D= 3

9) Asignación E= 3

 Solución 1
  A : 2
  B : 2
  C : 1
  D : 3
  E : 3
```

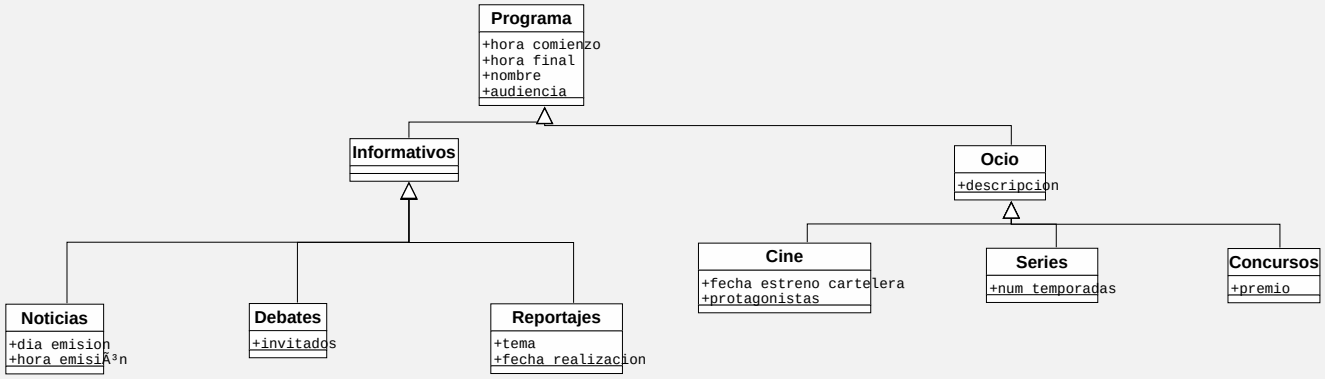
4.1 Problemas solucionados

8. El ministerio de telecomunicaciones, deseoso de poder manejar de manera sencilla y eficiente la información correspondiente a todos los medios de comunicación que emiten su señal al aire, quiere usar un sistema de frames para organizarla.

   La información que se desea organizar es la siguiente: Actualmente dos tipos de medios informativos emiten su información al espacio radioeléctrico, emisoras de radio y de televisión. Los programas que emiten se pueden clasificar esencialmente en dos, los programas de entretenimiento (variedades, concursos, cine, series, ...) y los programas de información (informativos, debates, reportajes, ...). Estos programas están asignados a las diferentes franjas horarias que forman la parrilla de emisión. Las diferentes franjas horarias estan clasificadas segun el público al que van dirigidos los programas (prime-time, infantil-juvenil, noctámbulos, ...) Cada emisora tiene su personal, que podríamos dividir en el personal técnico, que se encarga de que todo funcione correctamente (técnicos de sonido, cámaras, regidores, ...) y el personal no técnico, que es el que da la cara en los programas (presentadores, corresponsales, humoristas, etc.).

   a) Propón una representación estructurada del dominio, asignando entre uno y tres atributos a los conceptos que aparecen. Define el atributo **país de corresponsalía**, correspondiente a un corresponsal y el atributo **ranking** de un programa que indique su posición en los índices de audiencia según su tipo de programa.





El concepto *Franja horaria* podría ser un atributo de programa, pero tenerlo como un concepto nos facilitaría el saber que programas se emiten en cierta franja horaria tal como luego pide el apartado d.

Slot	País de corresponsalía	Ranking
Dominio	Corresponsal	Programa
Rango	String	Entero
Cardinalidad	1	1
Valor	—	—
Demon	—	—
Herencia	T:Si/U:no	T:Si/U:no

b) Define las relaciones entre personal y medio, entre personal y programa, entre medio y programa. Un de ellas debe ser compuesta.

Relación	Trabaja_en	Participa_en	Emite
Dominio	Personal	Personal	Medio
Rango	Medio	Programa	Programa
Cardinalidad	N	N	N
Inversa	Tiene_por_empleados (N)	Hecho_por (N)	Es_emitido_por (1)
Transitiva	no	no	no
Compuesta	Participa_en $\oplus$ Es_emitido_por	no	no
Demon	—	—	—
Herencia	—	—	—

Suponemos que el personal puede trabajar a la vez en varios medios.

c) Define un slot **estrella** en presentador, booleano, que sea cierto si el presentador está en un programa que se emite en la franja de prime-time. Añade lo que creas necesario.

Slot	Estrella
Dominio	Presentador
Rango	Booleano
Cardinalidad	1
Valor	—
Demon	if-needed es_prime_time?
Herencia	T:Si/U:no

Nos es necesaria una relación que represente la franaja horaria en la que es emitida un programa.

Relación	Emitido_en
Dominio	Programa
Rango	Franja
Cardinalidad	1
Inversa	Se_emiten (N)

```
funcion es_prime_time? ()
    retorna(F.Emitido_en==Prime-Time)
ffuncion
```

d) Queremos añadir a los medios un slot ranking que nos indique su posición en función del ranking de sus programas informativos. Usando únicamente las relaciones que hemos definido ¿podemos usar herencia? Justifica la respuesta

La herencia solo nos permite heredar valores, no podemos hacer ningún cálculo. Si una emisora tiene varios informativos obtendríamos varios valores que no tendríamos forma de combinar. Tenemos por lo tanto un problema de herencia mútiple.

Para poder obtener el valor de este slot deberíamos hacer especialización del slot **ránking** en el frame **medio** que tuviera un demon `if-needed` que obtuviera el ránking de todos los informativos del medio e hiciera el cálculo correspondiente.

e) Implementa un método parrilla, que nos liste para una determinada franja horaria el título de los programas que emite un medio informativo, junto al nombre de las personas que intervienen en él. ¿Dónde deberemos colocar este método?

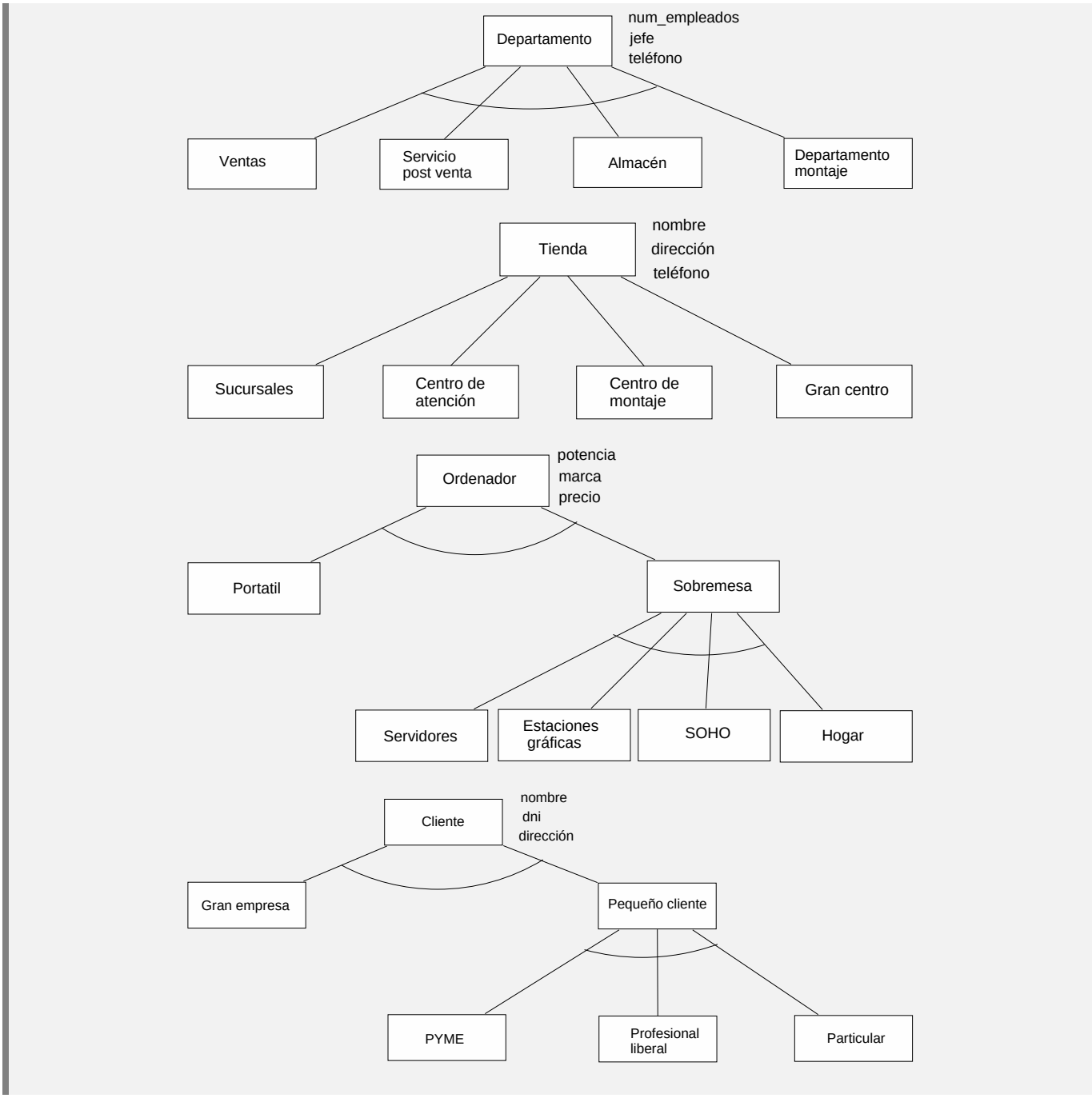
Este método lo podemos colocar en franja-horaria o en medio. El número de programas que tiene un medio en principio debería ser menor que el número de programas que se emiten en cierta franja horaria, por lo que quizas sea más eficiente colocarlo en medio.

```
funcion parrilla (string franja, string medio) retorna lista
    lista LP=∅
    lista LPers,LNom
    lista LMedios=Medio.tiene_por_instancias
    medio Medio

    para M en LMedios hacer
        si M.nombre==medio entonces Medio=M fsi
    fpara

    Progs= Medio.emite
    para P en Progs hacer
        si (P.Emitido_en).nombre==franja entonces
            LPers=P.hecho_por
            LNom=∅
            para Pe en LPers hacer
                LNom=LNom∪Pe.nombre
            fpara
            LP=LP ∪ (P.nombre, LN)
        fsi
    fpara
    retorna(LP)
ffuncion
```

11. Una gran cadena de distribución de ordenadores nos ha encargado que diseñemos para ella una representación del conocimiento que usa su organización de manera que pueda construir un sistema que lo utilice. Esta cadena está organizada en tipos de departamentos que pueden estar ubicados en las diferentes tiendas que tiene. Tenemos los departamentos de ventas, servicio post venta, almacén, y departamento de montaje. Las tiendas están organizadas en sucursales (sólo tienen departamento de ventas), centros de atención (sólo tienen servicio post venta), centros de montaje (sólo tienen almacén y departamento de montaje) y grandes centros (sólo tienen departamento de ventas y servicio post venta). Los ordenadores que vende la cadena se clasifican en portátiles y de sobremesa, éstos a su vez se clasifican en cuatro categorías: servidores, estaciones gráficas, SOHO y hogar.

    Esta cadena tiene dos tipos de clientes: las grandes empresas y los pequeños clientes, que incluyen las pyme, los profesionales liberales y los particulares.

    a) Propón una representación estructurada del dominio. Especifica al menos tres atributos para cada uno de los conceptos principales. Especifica completamente el atributo **número de empleados** en departamento y **potencia** en ordenador.



Slot	Número de empleados	Potencia
Dominio	departamento	ordenador
Rango	entero	real
Cardinalidad	1	1
Valor	—	—
Demon	—	—
Herencia	T:Si/U:no	T:Si/U:no

b) Define la relación **vende** entre departamento de venta y cliente, la relación **encarga** entre departamento de venta y de montaje, la relación **monta** entre departamento de montaje y ordenador y la relación **elige** entre cliente y ordenador.

Relación	Vende	Encarga	Monta
Dominio	D. Ventas	D. Ventas	D. montaje
Rango	Cliente	D. Montaje	Ordenador
Cardinalidad	N	N	N
Inversa	Compra (N)	Recibe_encargo (N)	Es_montado_por (1)
Transitiva	no	no	no
Compuesta	no	no	no
Demon	—	—	—
Herencia	—	—	—

Relación	Elige
Dominio	Cliente
Rango	Ordenador
Cardinalidad	N
Inversa	Es_elegido_por (1)
Transitiva	no
Compuesta	Compra $\oplus$ Encarga $\oplus$ Monta
Demon	—
Herencia	—

c) Define la relación **ubicado** entre departamento y tienda, esta relación ha de comprobar que el tipo de tienda acoge a los departamentos correctos. Define la relación **almacena** entre almacén y ordenador de manera que no se exceda la capacidad de almacenaje del almacén y que el ordenador se almacene en el mismo centro de montaje en el que el ordenador es montado. Define lo que sea necesario.

Relación	Ubicado
Dominio	Departamento
Rango	Tienda
Cardinalidad	1
Inversa	Tiene_ubicados (N)
Transitiva	no
Compuesta	no
Demon	if-added comprueba_departamentos
Herencia	—

accion comprueba_departamentos
si (instancia_de(R,Sucursal) y no (instancia_de?(D,Ventas)
entonces **error!**
si (instancia_de(R,Centro Atencion) y no instancia_de?(D,Serv Postventa)
entonces **error!**
si (instancia_de(R,Centro Montaje) y no (instancia_de?(D,Almacen) o instancia_de?(D,Montaje))
entonces **error!**
si (instancia_de?(R,Gran Centro) y no (instancia_de?(D,Ventas) o instancia_de?(D,Serv Postventa))
entonces **error!**
faccion

Relación	Almacena
Dominio	Almacen
Rango	Ordenador
Cardinalidad	N
Inversa	Esta_almacenado (1)
Transitiva	no
Compuesta	no
Demon	if-added comprueba_almacenaje
Herencia	—

Necesitamos un slot en almacen que nos diga su capacidad y otro que nos diga la cantidad de ordenadores que hay almacenados

```
accion comprueba_almacenaje
    si D.cantidad_almacenada + 1 > D.capacidad
    entonces error!
    si (((R.Es_montado_por).ubicado!=D.ubicado)
    entonces error!
faccion
```

d) Queremos que las tiendas hereden el slot **número de empleados** de los departamentos que tiene ubicados define lo necesario para que esto sea posible. ¿Es la herencia la mejor solución?

Dado que una tienda puede tener varios departamentos no podemos obtener el total de la tienda utilizando herencia ya que tendríamos un problema de herencia múltiple, lo mas sencillo es redefinir el slot numero de empleados en tienda y hacer un demon `if-needed`.

Slot	Número de empleados
Dominio	tienda
Rango	entero
Cardinalidad	1
Valor	—
Demon	if-needed calcula_empleados
Herencia	T:Si/U:no

```
funcion calcula_empleados ()
    entero n=0
    LDep=F.Tiene_ubicados
    para D en LDep hacer
        n=n+D.num_empleados
    fpara
    retorna(n)
ffuncion
```

e) Define el método **pedidos** que liste los ordenadores de un determinado tipo vendidos, indicando los nombres de los clientes a los que se han vendido y el centro de montaje en el que se ha encargado el montaje de cada ordenador. Ubica adecuadamente el método. Define lo que creas necesario para que se pueda implementar el método ¿Es heredable este método?

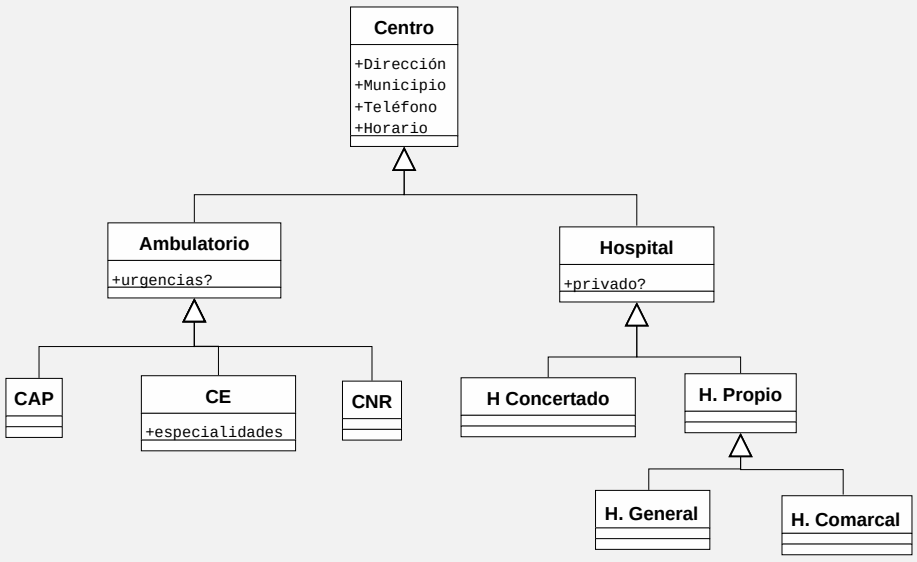
El método debería ubicarse en el concepto **ordenador**. El método no sería heredable ya que al tener el tipo de ordenador como parámetro sería incoherente invocarlo desde alguna de las subclases.

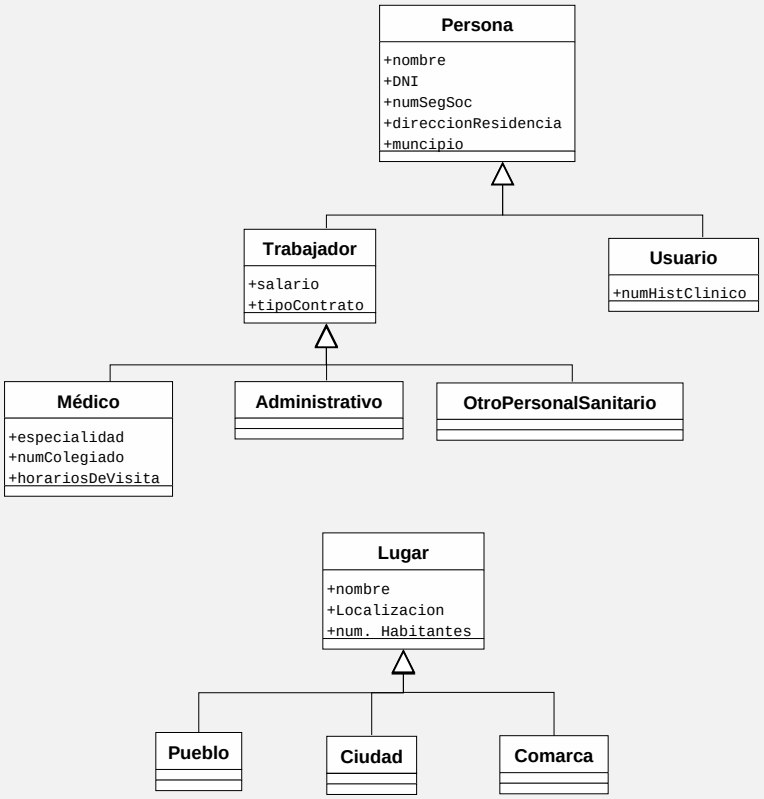
```
funcion pedidos (string tipo) retorna lista
        lista LP=∅
        lista LOrd

        si tipo==portatil entonces LOrd=Portatil.tiene_por_instancias fsi
        si tipo==sobremesa entonces LOrd=Portatil.tiene_por_instancias fsi
        para O en LOrd hacer
            LP=LP ∪ (O, O.Es_elegido.nombre, O.Es_montado_por)
        fpara
        retorna(LP)
ffuncion
```

12. El Institut Català de la Salut (ICS) desea organizar toda la información relativa a sus competencias. Por un lado, desea representar la información de los centros: ambulatorios y hospitales. Existen tres tipos de ambulatorios: centro de asistencia primaria (CAP), centro de especialistas (CE) i centro no reconvertido (CNR). Algunos ambulatorios disponen de un servicio de urgencias y otros no. Los hospitales propios pueden ser generales o comarcales. Los hospitales concertados son hospitales privados con convenio de asistencia con el ICS.

    También desea representar información sobre personas tanto de sus trabajadores (médicos, asistentes sociales, ayudantes sanitarios, administrativos, ...) como de los usuarios de la institución. Igualmente necesita representar información geográfica: pueblos, ciudades, comarcas, ...

    a) Propón una representación estructurada del dominio. Especifica al menos tres atributos para cada uno de los conceptos principales. Especifica completamente el atributo **especialidades** de los CE, el atributo **especialidad** de médico y el atributo booleano **privado** de hospital.





Slot	Especialidades	Especialidad	Privado
Dominio	C. Especialistas	Medico	Hospital
Rango	String	String	Booleano
Cardinalidad	N	1	1
Valor	—	—	—
Demon	—	—	—
Herencia	T:Si/U:no	T:Si/U:no	T:Si/U:no

b) Define la relación **referente** de hospital respecto de los ambulatorios de los cuales es referencia, las relaciones de **pertenencia** entre pueblo/ciudad y comarca y entre hospital comarcal y comarca y la relación de **habitante** entre persona y pueblo/ ciudad. A partir de todo esto, define la relación de **adscripción** entre pueblo y hospital comarcal y la de **ingreso-por-defecto** entre habitante y hospital comarcal.

Relación	referente	Pertenece_lugar	Pertenece_Hospital
Dominio	Hospital	Pueblo, Ciudad	H. Comarcal
Rango	Ambulatorio	Ambulatorio	Comarca
Cardinalidad	N	1	1
Inversa	Tiene_por_referente (1)	Tiene_por_lugar (N)	Tiene_por_hospital (1)
Transitiva	no	no	no
Compuesta	no	no	no
Demon	—	—	—
Herencia	—	—	—

Relación	Habitante	Adscripción
Dominio	Persona	Pueblo
Rango	Pueblo, Ciudad	H. Comarcal
Cardinalidad	1	1
Inversa	Tiene_por_habitantes (N)	Tiene_adscritos (N)
Transitiva	no	no
Compuesta	no	Pertenece_lugar ⊕ Tiene_por_hospital
Demon	—	—
Herencia	—	—

Relación	Ingreso_por_defecto
Dominio	Habitante
Rango	H. Comarcal
Cardinalidad	1
Inversa	Recibe_Pacientes (N)
Transitiva	no
Compuesta	habitante ⊕ Adscripción
Demon	—
Herencia	—

Definimos dos relaciones de pertenencia para poder realizar la composición en la relación adscripción.

c) Define la relación de **asignado** entre médico y centro/s donde trabaja de forma que se compruebe para los CE que se está asignando un médico cuya especialidad es una de las disponibles en el CE.

Relación	Asignado
Dominio	Médico
Rango	Centro
Cardinalidad	N
Inversa	Tiene_asignados (1)
Transitiva	no
Compuesta	no
Demon	if-added comprueba_especialidad
Herencia	—

```
accion comprueba_especialidad
    si CE?(R) entonces
        si D.especialidad ∉ R.especialidades
        entonces error!
        fsi
    fsi
faccion
```

d) Se desea que el atributo **privado** para un médico se infiera del hospital en el cual trabaja, pero no todos los médicos trabajan en un hospital. ¿Qué modificaciones debes hacer para que esto sea posible? ¿Sirven los mecanismos de herencia?

Tenemos definida la relación **asignado** con rango **centro**, esto supone dos problemas, por un lado, no todas las instancias con las que está relacionado un médico tienen el slot que queremos, por otro lado, tenemos un problema de heréncia múltiple ya que un médico puede estar asignado a vários hospitales.
Debemos por tanto resolverlo mediante un demon redefiniendo el slot privado en el frame médico:

Slot	Privado
Dominio	Medico
Rango	Booleano
Cardinalidad	1
Valor	—
Demon	if-needed funcion calcula_privado
Herencia	T:Si/U:no

Supondremos que un médico es privado si está asignado a algún hospital privado.

```
funcion calcula_privado retorna booleano
    centro LCentros=F.asignado
    booleano privado=falsp

    para C en LCentros hacer
        si Instancia_de(C,Hospital)
            privado=privado o C.privado
        fsi
    fpara
    retorna(privado)
ffuncion
```

e) Define el método **especialistas** que liste, para un ambulatorio dado, todos los médicos de una cierta especialidad que trabajan en el hospital de referencia del ambulatorio. Para cada médico se desea obtener su nombre, su número de colegiado y sus horarios de visita. Ubica adecuadamente el método. Define lo que creas necesario para que se pueda implementar el método ¿Es heredable este método?

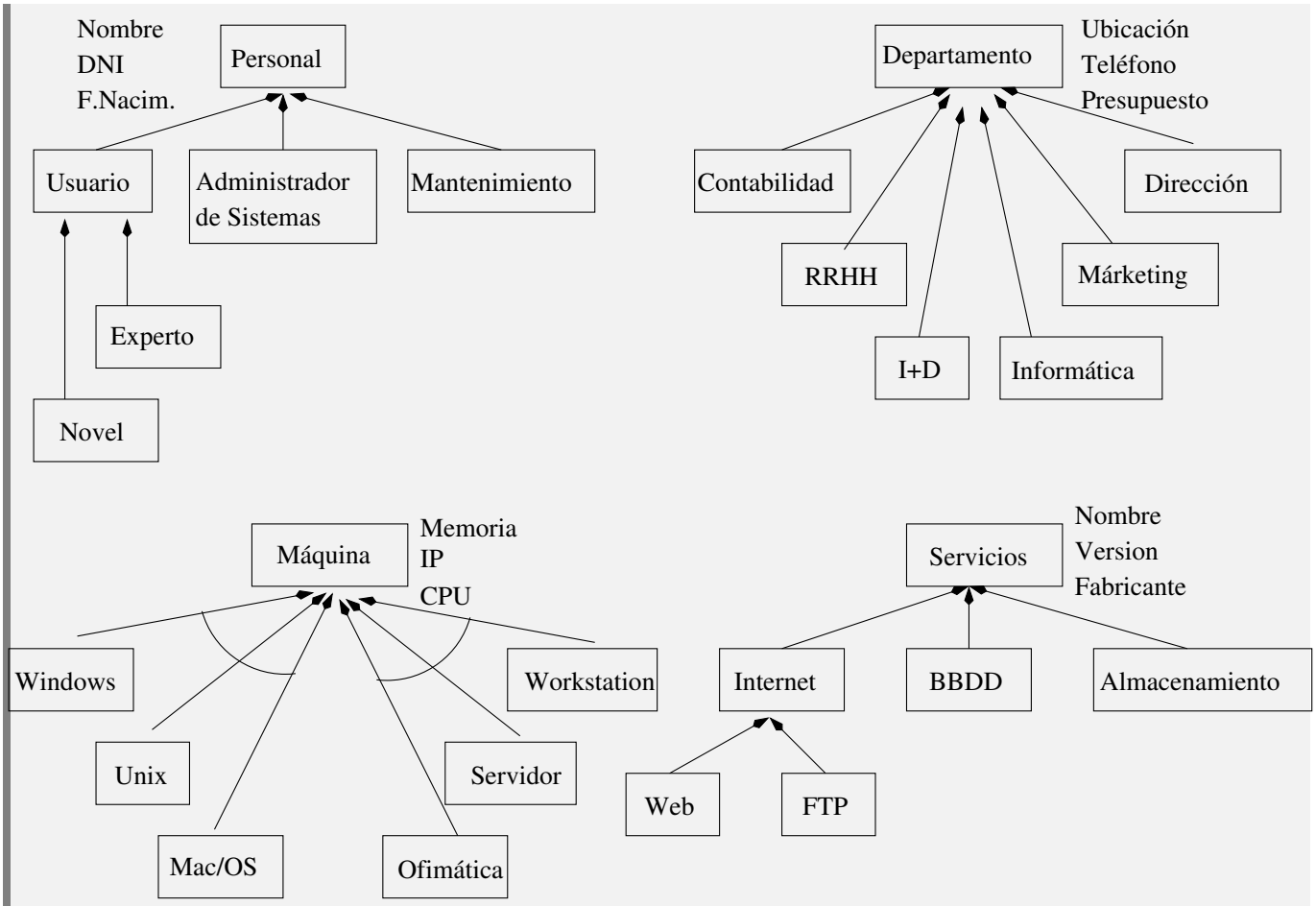
Nos harán falta añadir los slots que nos piden en el método en el frame **médico**.
El método lo ubicaremos en el frame **ambulatorio** que es desde donde podemos acceder al hospital de referencia del ambulatorio.

```
funcion medicos_especialidad (string especialidad) retorna lista
    lista LP=∅
    hospital H = F.Tiene_por_referente
    medico LMedicos=H.tiene_asignados

    para M en LMedicos hacer
        si M.especialidad==especialidad entonces
            LNom=LNom∪(M.nombre,M.num_colegiado,M.horas_visita)
        fsi
    fpara
    retorna(LP)
ffuncion
```

16. Una gran empresa multinacional desea organizar la información relativa a la gestión de los recursos informáticos de su organización. La descripción de su problemática es la siguiente: La empresa cuenta con personal que utiliza y mantiene los diferentes recursos, este personal se puede clasificar en: usuarios (expertos o noveles), administradores de sistemas y personal de mantenimiento. El personal está organizado en departamentos (contabilidad, recursos humanos, investigación y desarrollo, marketing, informática, dirección), cada departamento tiene una ubicación.

    Para organizar el parque informático se piensa en contemplar una clasificación según el sistema operativo de la máquina (Windows, Unix, Mac/OS) y según el uso que se hace (workstation, servidor, ofimática). Estas máquinas proveen diferentes servicios, como por ejemplo servicios de Internet (servicio de web, de ftp) servicio de base de datos, servicio de almacenamiento, ...

    a) Propón una representación estructurada del dominio. Especifica al menos tres atributos para cada uno de los conceptos principales y caracteriza completamente tres de ellos



Slot	Nombre	Ubicación	IP
Dominio	Personal	Departamento	Máquina
Rango	string	string	string
Cardinalidad	1	1	1
Valor	—	—	—
Demon	—	—	—
Herencia	T:Si/U:no	T:Si/U:no	T:Si/U:no

    b) Las máquinas de la empresa se pueden asignar directamente a usuarios o a departamentos, en ambos casos el uso de la maquina es en exclusiva. En el caso de máquinas asignadas a departamentos, estas pueden ser usadas por muchos usuarios, pero solo del departamento, caracteriza

estas relaciones y añade lo que sea necesario a la representación.

La exclusividad de la asignación la garantiza la cardinalidad.

Relación	Asignada
Dominio	Máquina
Rango	Usuario, Departamento
Cardinalidad	1
Inversa	Tiene_asignadas (N)
Transitiva	no
Compuesta	no
Demon	if-added comprueba_uso
Herencia	—

Nos hace falta una relacion `uso` que nos permita saber quién está usando la máquina asignada y una relación entre los usuarios y su departamento. Asumimos que un usuario puede utilizar varios ordenadores y que un usuario solo puede pertenecer a un departamento.

Relación	usa	pertenece
Dominio	Usuario	Usuario
Rango	Máquina	Departamento
Cardinalidad	N	1
Inversa	es_usado_por(N)	esta_integrado_por (N)
Transitiva	no	no
Compuesta	no	no
Demon	—	—
Herencia	—	—

```
accion comprueba_uso
    si Departamento?(R) entonces
        LU= D.es_usado_por
        para U en LU hacer
            si U.pertenece!=R entonces error!
        fpara
    fsi
faccion
```

c) Las máquinas están ubicadas en el departamento en el que están asignadas o en el departamento al que pertenece la persona que la tiene asignada. Añade a la representación y/o modifica lo necesario para que esta información se pueda obtener mediante el mecanismo de herencia.

En este caso la herencia nos debería permitir consultar el slot `Ubicación` del frame `departamento` en el frame `máquina`. Para obtenerlo en el caso de que la muina este asigna da a un departamento, deberemos permitir que pueda ser heredado por el frame `máquina` a través de la relaciones de usuario, y heredarlo a traver de la relación `Asignada`. Para obtenerlo en el caso de que la máquina este asignada a un usuario tendremos que permitir que también se herede a través de la relación `pertenece`

Slot	Ubicación
Dominio	Departamento, Máquina, Usuario
Rango	string
Cardinalidad	1
Valor	—
Demon	—
Herencia	T:Si/U:si

Relación	asignada	pertenece
Dominio	Máquina	Usuario
Rango	Usuario, Departamento	Departamento
Cardinalidad	1	1
Inversa	Tiene_asignadas (N)	esta_integrado_por (N)
Transitiva	no	no
Compuesta	no	no
Demon	—	—
Herencia	Ubicación	Ubicación

d) Los servidores son proveedores de los diferentes servicios descritos, que son instalados por el personal de administración. Añade lo necesario a la representación para que se pueda obtener para una máquina específica la lista de las personas que han instalado los diferentes servicios que provee esa máquina. ¿que diferencia habría respecto de usar un demon o un método? ¿y si quisiéramos sólo la información de un tipo de servicio concreto?

Nos hará falta una relación entre `servicio` y `administrador de sistemas` y entre `servidor` y `servicios`

Relación	instala	provee
Dominio	Administrador	Servidor
Rango	Servicio	Servicio
Cardinalidad	N	N
Inversa	es_instalado_por(1)	es_dado_por (1)
Transitiva	no	no
Compuesta	no	no
Demon	—	—
Herencia	—	—

Podríamos hacer indistintamente un demon `if needed` en `servidor` o un método, pero en el caso de querer un tipo de servicio concreto solo podríamos usar un método.

```
función lista_instaladores retorna lista
    lista LS=F.provee
    lista LAd=∅
    para S en LS hacer
        LAd=LAd∪S.es_instalado_por
    fpara
    retorna(LAd)
ffunción
```

e) Supongamos que queremos obtener de la representación un listado de todos los usuarios de cierto tipo que usan una máquina con cierto sistema operativo, ¿que le deberíamos añadir a la repre-

sentación para que fuera posible? Caracteriza y/o programa el slot, demon o método que sea necesario.

En este caso nos será necesario un método ya que tenemos parámetros. La ubicación mas adecuada para este método es el frame `Usuario`. Este método no será heredable.

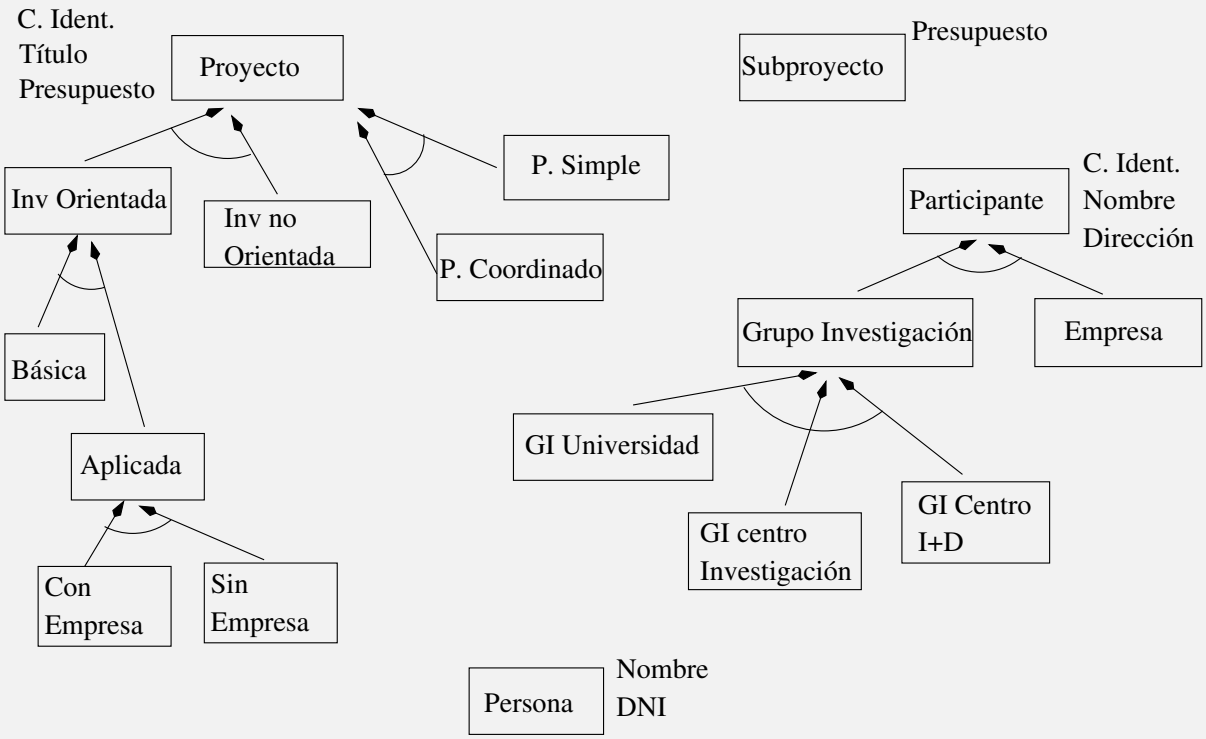
```
función lista_usuarios (string tipo_usuario, string SO) retorna lista
    lista LUs
    Booleano Ok
    si tipo_usuario=Experto entonces
        LUs= Experto.tiene_por_instancias
    si no
        LUs= Novel.tiene_por_instancias
    fsi
    para U en LUs hacer
        LMa= U.usa
        Ok=false
        para M en LMa hacer
            ok=ok o M.SO==SO
        fpara
        si Ok entonces LUs=LUs∪U
        fsi
    fpara
    retorna(LUs)
ffunción
```

17. El Ministerio de Ciencia y Tecnología desea organizar toda la información relativa a los proyectos de investigación que financia. El Ministerio distingue entre dos tipos de proyectos: investigación orientada e investigación no orientada. Los primeros, a su vez, se dividen en proyectos de investigación básica y proyectos de investigación aplicada. En el caso de la investigación aplicada hay dos modalidades: con participación opcional de empresas y con participación obligatoria de empresas. Los proyectos tienen un código de identificación, un título, un presupuesto, unos objetivos, etc. Los proyectos pueden ser presentados por un único participante (proyecto simple) o por varios (proyecto coordinado). Los proyectos simples tienen un participante como responsable. Los proyectos coordinados constan de varios proyectos simples (los subproyectos), cada uno con un participante como responsable. Además, uno de estos será el coordinador del proyecto coordinado.

    Los participantes pueden ser Empresas y grupos de investigación pertenecientes a Universidades, Centros públicos/privados de investigación, Centros públicos/privados de I+D. Las empresas siempre tienen que participar en proyectos en colaboración con grupos de investigación. Las empresas no pueden actuar como coordinador de proyecto, salvo en el caso de los proyectos de investigación aplicada con participación obligatoria de empresas donde el coordinador ha de ser forzosamente una empresa.

    Cada grupo de investigación cuenta con un investigador principal y varios investigadores colaboradores. Cada empresa cuenta con un representante y varias personas adscritas. Cada grupo y cada empresa tienen un código de identificación.

    a) Propón una representación estructurada del dominio. Asigna al menos tres atributos a cada uno de los conceptos principales. Caracteriza las relaciones entre proyecto coordinado y subproyectos, entre proyecto simple y participante responsable, entre proyecto coordinado y participante coordinador y entre persona y grupo de investigación al que pertenece. Caracteriza completamente el slot código de identificación.



Relación	Formado_por	Responsable	Coordinador
Dominio	P. Coordinado	Participante	Participante
Rango	Subproyecto	P. Simple	P. Coordinado
Cardinalidad	N	N	N
Inversa	Forma_parte_de (1)	Tiene_por_responsable (1)	Tiene_por_coordinador (1)
Transitiva	no	no	no
Compuesta	no	no	no
Demon	—	—	—
Herencia	—	—	—

Relación	Pertenece
Dominio	Persona
Rango	Participante
Cardinalidad	1
Inversa	Integrado_por (N)
Transitiva	no
Compuesta	no
Demon	—
Herencia	—

Slot	C. Ident
Dominio	Proyecto, Participante
Rango	string
Cardinalidad	1
Valor	—
Demon	—
Herencia	T:Si/U:no

b) Incorpora lo que sea necesario a las relaciones entre proyecto y participante responsable/coordi-

nador que compruebe que el responsable **HA** de ser una empresa para el caso de proyectos de investigación aplicada con participación obligatoria de empresas y **NO** ha de serlo en cualquier otro caso.

Relación	Responsable	Coordinador
Dominio	Participante	Participante
Rango	P. Simple	P. Coordinado
Cardinalidad	N	N
Inversa	Tiene_por_responsable (1)	Tiene_por_coordinador (1)
Transitiva	no	no
Compuesta	no	no
Demon	if-added comprueba_responsable	if-added comprueba_coordinador
Herencia	—	—

Debemos comprobar que las empresas no sean responsables en proyectos simples (no pueden ir solas) y que en proyectos coordinados solo sean coordinadores de los proyectos que exijan una empresa

```
acción comprueba_responsable
    si Empresa?(R) entonces error
facción
```

```
acción comprueba_coordinador
    si Empresa?(R) entonces
        si no con_empresa(D) entonces error
    fsi
facción
```

c) Un proyecto de cualquier tipo se considera de alto nivel si el participante responsable o coordinador lo es. ¿Puedes establecer mediante mecanismos de herencia esta inferencia? En caso afirmativo, propónlos. En caso negativo, propón otro mecanismo para poder realizar la inferencia.

Se puede obtener esta información por herencia creando el slot `alto_nivel` en participante y heredándolo a través de las relaciones `responsable` y `coordinador`.

Slot	alto_nivel
Dominio	Participante
Rango	booleano
Cardinalidad	1
Valor	—
Demon	—
Herencia	T:Si/U:si

Relación	Responsable	Coordinador
Dominio	Participante	Participante
Rango	P. Simple	P. Coordinado
...	...	...
Herencia	alto_nivel	alto_nivel

d) Un proyecto coordinado se considera "conflictivo" si el número de participantes es superior a seis o bien si al menos tres de los participantes son empresas, sea cual sea el número total de participantes. Establece el mecanismo adecuado que permita consultar si un proyecto coordinado es conflictivo o no.

En este caso necesitaremos un slot en el frame `P. coordinado` que calcule si el proyecto es conflictivo o no mediante un demon. Asumiremos que la relacion `responsable` también puede tener como rango los subproyectos.

Slot	conflictivo
Dominio	P. Coordinado
Rango	booleano
Cardinalidad	1
Valor	—
Demon	if-needed calcula_conflictivo
Herencia	T:Si/U:si

```
función calcula_conflictivo
        entero emp=0
        lista LSub
        si card(F.Formado_por)>6 entonces retorna (cierto)
        si no   LSub= F.Formado_por
            para S en LSub hacer
                si Empresa?(S.Tiene_por_responsable ) entonces emp++
            fpara
            si emp>2 entonces retorna (cierto)
        fsi
        retorna (falso)
función
```

e) Diseña un método con un parámetro que permita listar todos los proyectos de investigación orientada que sean de un determinado tipo de los tres posibles. Por cada proyecto, hay que mostrar su código, título, presupuesto, participante responsable/coordinador, investigador principal (o representante) y, en el caso de los coordinados, la identificación de todos los grupos/empresas participantes. Añade lo que sea necesario. Sitúa adecuadamente este método. ¿Es heredable?

El método lo deberemos colocar en `P. Inv Orientada` y no será heredable. Nos serán necesarias las relaciones `investigador_principal` y `representante` entre `persona` y `grupo de investigacion` y `empresa`

```
función lista_proyectos (string tipo) retorna lista
    lista LRes=∅
    lista LPar
    lista LProy
    opción
        caso (tipo=básica):
            LProy=Básica.tiene_por_instancias
        caso (tipo=Aplicada con empresa):
            LProy=Aplicada con empresa.tiene_por_instancias
        caso (tipo=Aplicada sin empresa):
            LProy=Aplicada sin empresa.tiene_por_instancias
```

```
                    fopción

                    para P en LProy hacer
                        opción
                            caso (P. Simple(P)):
                                Part= P.responsable
                                Pers= Part.investigador_principal
                                LRes=LRes∪ (P.código,P.Título,P.presupuesto,Part,Pers)

                            caso (P. Coordinado(P)):
                                Part= P.coordinador
                                opción
                                    caso (Grupo Investigacion?(Part)):
                                    Pers= Part.investigador_principal
                                    caso (Empresa?(Part)):
                                    Pers= Part.representante
                                fopción

                                LSub= P.Formado_por
                                LPar=∅
                                para S en LSub hacer
                                    LPar=LPar∪ S.Tiene_por_responsable.CIdent entonces emp++
                                fpara
                                LRes=LRes∪ (P.código,P.Título,P.presupuesto,Part,Pers,LPar)
                        fopción
                    fpara
                    retorna(LRes)
                función
```

5.1 Problemas solucionados

2. El departamento de planificación urbana de la ciudad de Urbópolis ha decidido utilizar un SBC para la planificación de los diferentes elementos de la ciudad.

   Para este departamento la ciudad está compuesta por unidades urbanas de tres tipos distintos. El primer tipo son las unidades comerciales que pueden corresponder a comercios mayoristas o comercios minoristas. El segundo tipo son las unidades de servicio al ciudadano tales como colegios públicos, bibliotecas, zonas verdes, polideportivos, hospitales, centros de asistencia primaria, plazas públicas, farmacias y oficinas de servicios públicos. Finalmente, el tercer tipo son las unidades residenciales que pueden corresponder a viviendas unifamiliares, edificios de pisos y manzanas de viviendas.

   Las unidades urbanas se pueden ubicar en espacios. Hay espacios de dos tipos. El primero son los solares que pueden clasificarse en bloques (capacidad para un solo edificio), manzanas (capacidad para varios edificios) y grandes áreas (capacidad para grandes instalaciones). El segundo tipo son los locales.

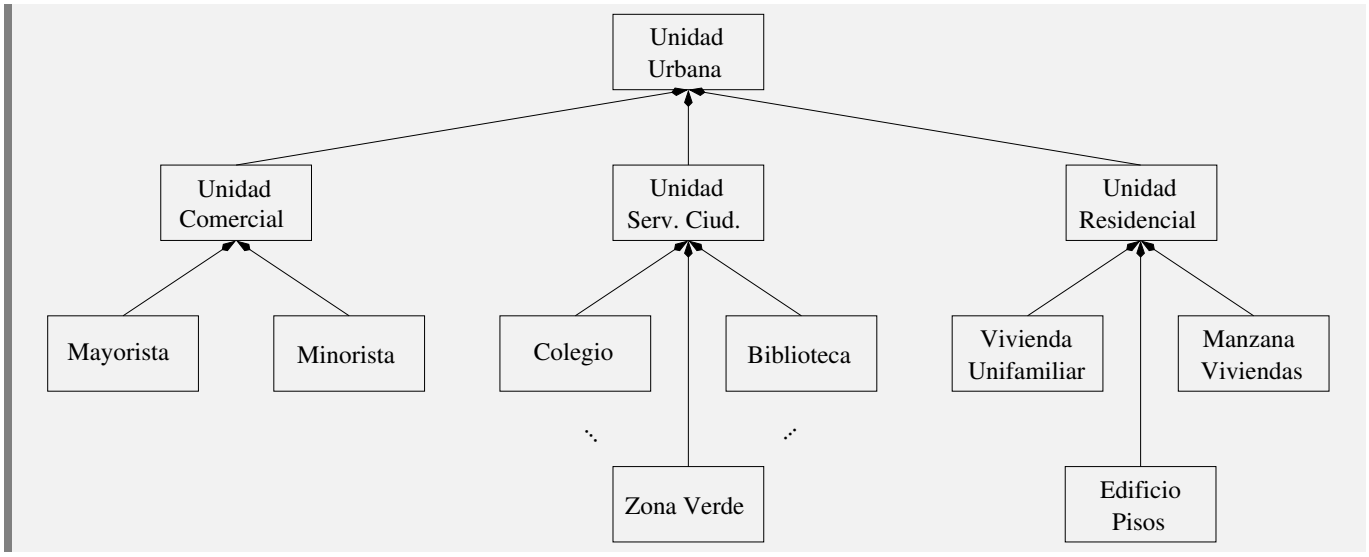
   El departamento de planificación tiene la información de todos los espacios que existen en la ciudad. También dispone de la información de qué espacios están vacantes en la actualidad y qué unidades urbanas se han asignado a los espacios ya ocupados. A cada espacio solo se le puede asociar una unidad urbana.

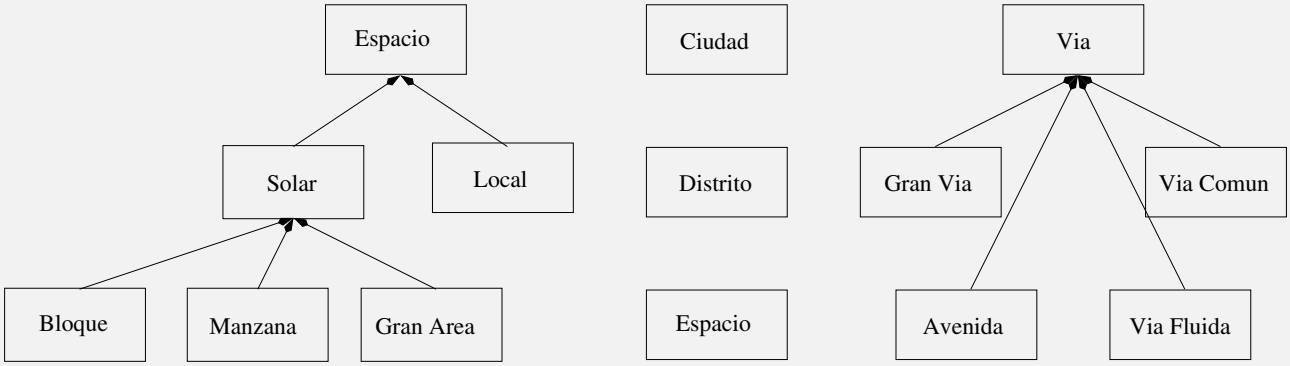
   Las unidades urbanas se encuentran dentro de barrios, que es la división más fina en la que el ayuntamiento agrupa las zonas de la ciudad. El barrio tiene información sobre su población, incluyendo su distribución por edades (niños, adultos, ancianos). Los barrios a su vez están agrupados en distritos.

   Otro elemento de la ciudad son las vías. Estas se pueden clasificar en grandes vías, avenidas, vías fluidas y vías comunes. Todo espacio tiene una serie de vías con las que está relacionado (aquellas que lo bordean y/o cruzan). Cada vía tiene información de la cantidad de coches por minuto que la recorren. Estas vías estarán relacionadas también con barrios y distritos.

   A) Preguntas de Ontologías/Frames

   a) Propón una representación estructurada del dominio descrito e identifica los atributos más relevantes.





- Atributos de Unid.Urbana: superficie, precio_m2, titularidad, tipo
- Atributos de Espacio: superficie, identificador_catastral
- Atributos de Via: densidad_trafico, sentido_circulacion, orientación
- Atributos de Barrio: habitantes, poblac_infantil, poblac_adulta, poblac_anciana
- Atributos de Distrito: nombre, numero_distrito

b) Define todas las relaciones de usuario que consideres necesarias y, en particular, especifica **ubicado_en** entre unidad urbana y espacio, **situado_en** entre espacio y barrio, **relacionado_con** entre espacio y vía.

Relaciones en principio necesarias: las que nos piden más las que indican relación entre Barrio, Distrito y Ciudad.

Relacion	UBICADO_EN	SITUADO_EN	RELACIONADO_CON
Dominio:	UNIDAD_URBANA	ESPACIO	ESPACIO
Rango:	ESPACIO	BARRIO	VIA
Cardinalid:	1	1	N
Inversa:	OCUPADO_POR (1)	FORMADO_POR (N)	TOCA (N)
Compuesta:	—	—	—
Transitiva:	—	—	—
Demons:	—	—	—
Herencia:	NO	NO	NO

La relación PARTE_DE la tenemos entre dos conjuntos de frames se puede definir de la siguiente manera:

Relación:	PARTE_DE
Dominio:	Barrio, Distrito
Rango:	Distrito, Ciudad
Cardinalid:	1
Inversa:	COMPUESTO_POR(N)
Compuesta:	—
Transitiva:	SI
Demons:	—
Herencia:	NO

c) Propón e implementa un mecanismo adecuado para obtener el total de la población infantil en la ciudad.

Poner un slot, Total_infantil, en el frame CIUDAD con un demon asociado if_needed, de modo que se recorran todos los distritos de la ciudad y para cada distrito todos sus barrios y se va acumulando la población infantil de cada uno de ellos. Ese valor acumulado será el que se asocie al slot Total_infantil.

Slot:	Total_infantil
Dominio:	CIUDAD
Rango:	entero
Cardinalidad:	1
Valor:	—
Demons:	if_needed Calcular_total( ) return entero
Herencia:	TAX=si, USU=no

```
funcion  Calcular_total() return entero
  /* F es la ciudad */
  /* LD es la lista de todos los barrios de la ciudad */
  LD:= F.COMPUESTO_POR
  total:=0
  para cada d en LD hacer
     /* LB es la lista de todos los barrios del distrit. d */
     LB:=d.COMPUESTO_POR
     acum:=0
     para cada b en LB hacer
        acum:=acum+ b.poblacion_infantil
     fpara
     total:=total+acum
  fpara
  return total
ffuncion
```

d) Define todo lo necesario para determinar a qué distrito pertenece una unidad urbana y a qué barrios pertenece una vía ¿Es posible asociar números de distrito a barrios y vías usando el mecanismo de herencia? Justifica tu respuesta.

¿a qué distrito pertenece una unidad urbana?

Hay dos posibilidades:

1) De UNIDAD URBANA y navegando por la relación UBICADO_EN, llegar a ESPACIO y, una vez tenemos la instancia de ESPACIO, navegando por la relación SITUADO_EN llegar al BARRIO. Situados en el BARRIO y a través de la relación PARTE_DE obtenemos el distrito al que pertenece la unidad urbana.

2) Se puede definir una nueva relación directa entre UNIDAD URBANA y DISTRITO. Esta nueva relación se puede obtener por composición de 3 ya existentes.

Relación:	PERTENECE_DISTRITO
Dominio:	UNIDAD URBANA
Rango:	DISTRITO
Cardinalidad:	1
Inversa:	DISTRITO_URBANIZADO(N)
Compuesta:	SI (PARTE_DE $\otimes$ SITUADO_EN $\otimes$ UBICADO_EN(x), x unidad urbana)
Transitiva:	—
Demons:	—
Herencia:	NO

¿a qué barrios pertenece una via?

De la definición de las relaciones que hemos hecho, se tiene que una via puede estar relacionada con más de un espacio y cada uno de ellos puede pertenecer a un barrio distinto. Haremos lo mismo que en la pregunta anterior.

1) Navegar por las relaciones existentes para obtener los barrios y de ahí llegar a los barrios.

2) Igual que en la pregunta anterior, podemos definir una nueva relación resultado de componer dos ya existentes: TOCA y SITUADO_EN.

Relación:	VIA_EN_BARRIO
Dominio:	VIA
Rango:	BARRIO
Cardinalidad:	N
Inversa:	BARRIO_CON_VIAS (N)
Compuesta:	SI (SITUADO_EN $\otimes$ TOCA(x), con x una via)
Transitiva:	—
Demons:	—
Herencia:	NO

¿Es posible asociar números de distrito a barrios y vías usando herencia?

Se deduce que un DISTRITO tendrá un slot numero_distrito.

Aparecen dos posibilidades:

1) Como un barrio pertenece a un único distrito, un barrio puede heredar a través de la relación COMPUESTO_POR el slot numero_distrito.

2) Gracias a la relación compuesta definida entre VIA y BARRIO podríamos pensar en que una vía también puede heredar el numero_distrito que ha heredado a su vez BARRIO pero NO puede ser porque una via puede pertenecer a barrios distintos y entonces heredaría valores diferentes (conflicto de herencia múltiple).

Para implementar A, que es lo único que se puede hacer con herencia, hay que redefinir el slot numero_distrito y modificar la herencia de la relación COMPUESTO_POR

Slot:	numero_distrito
Dominio:	DISTRITO, BARRIO
Rango:	entero
Cardinalidad:	1
Herencia:	TAX=si, USU=si

Relación:	COMPUESTO_POR
Dominio:	DISTRITO
Rango:	BARRIO
Cardin.:	N
Herencia:	SI (numero_distrito)

e) Define lo necesario para obtener un listado de todos los espacios vacantes de la ciudad y de todos los ocupados indicando qué uso se les ha asignado.

Pondremos un método en ESPACIO y no heredable por las instancias. Devolverá dos listas, una con los espacios vacantes y otra con los ocupados y el uso que tienen.

```
método función listar() retorna lv, lo es lista  {no heredable}
  lv, lo:= lista_vacia
  LE:=F.tiene_por_instancia  /*LE es la lista de todos los espacios */
    para cada elem en LE hacer
     si OCUPADO_POR(elem) entonces
      x:=elem.OCUPADO_POR
      opcion
```

```
            instancia?(x, U.COMERCIAL):
              lo:=añadir(lo, <elem.identificador_catastral, x.tipo>)
            /* hacer esto para cada tipo de uso*/
           fopcion
          si_no lv:=añadir(lv, <elem.identificador_castastral,'vacante'>)
          fsi
         fpara
       retorna lv, lo
     ffuncion
```

f) Define lo necesario para obtener la relación de todas las unidades urbanas de un tipo dado ubicadas en un distrito.

Método en DISTRITO heredable por las instancias con un parámetro que es el tipo de uso de la unidad urbana. El método ha de obtener todas las instancias de UNIDAD_URBANA relacionadas con el DISTRITO a través de la relación compuesta DISTRITO_URBANIZADO y para cada una de ellas hay que averiguar si son instancias del tipo que nos pasan como parámetro. Si lo son se añaden a la lista y si no, no se añaden.

B) Preguntas de IC

a) El primer objetivo del departamento de planificación es hacer una clasificación de los espacios vacantes de manera que se pueda tener un catálogo que indique qué tipos de usos son adecuados para cada una de ellos. Suponemos que a cada espacio solo se le puede asociar una unidad urbana.

Por el conocimiento de los ingenieros urbanísticos, se sabe que las unidades comerciales solo pueden ser ubicadas en locales. Lo mismo sucede con las oficinas de servicios públicos y farmacias que solo pueden ubicarse en locales. Las bibliotecas se pueden colocar en locales si la población del barrio no es muy grande o ya existen otras bibliotecas en el barrio. En caso contrario se ubican en bloques.

Respecto a las unidades residenciales, las viviendas unifamiliares y los edificios de pisos solo se pueden ubicar en bloques y las manzanas de viviendas solo se pueden ubicar en manzanas.

Para el resto unidades de servicio al ciudadano, los colegios, centros de asistencia primaria y plazas solo pueden ubicarse en manzanas y los polideportivos y parques solo pueden hacerlo en grandes áreas.

Para decidir el uso de un espacio se tienen en cuenta además otras características de la zona donde está ubicado ese espacio. Por ejemplo, se tiene en cuenta la densidad de población del barrio (*alta*, *media*, *baja*) globalmente o por tipos de población. También se estima la densidad del tráfico del barrio (*alta*, *media*, *baja*) respecto al número de vías que pasan por él y su densidad.

También se tienen en cuenta las diferentes proximidades (*al lado*, *cerca*, *media*, *lejos*) de las unidades urbanas ya ubicadas en los espacios, tanto a nivel de barrio como a nivel de distrito. Finalmente también se tienen en cuenta el tipo de vías con las que está relacionado el espacio y la densidad de su tráfico.

Por ejemplo, los colegios se deben ubicar en zonas de *alta* densidad de población infantil, donde haya al menos *cerca* bibliotecas y zonas verdes como máximo a *media* distancia del colegio. La densidad del tráfico de las vías *cercanas* ha de ser baja. Todo colegio debe estar a distancia *media* o *lejos* de cualquier otro colegio.

Los centros de asistencia primaria deben ubicarse preferiblemente en zonas de *alta* densidad de población infantil y/o anciana, *cerca* de alguna farmacia, con una densidad de tráfico alrededor *media*, *cerca* de alguna vía fluida y a una distancia *lejos* de otros centros de asistencia primaria.

El problema que se plantea es un problema de análisis y se puede resolver mediante clasificación heurística. Determina cómo se ubicarían los pasos de resolución de este problema en cada una de las fases de esta metodología. Da diferentes ejemplos de reglas para cada una de las fases que muestren como se llegaría a asociar un colegio a un espacio.

Como comenta el enunciado, se trata de un problema de análisis. En esta caso hemos de obtener una asignación de unidades urbanas a espacios.

Esto quiere decir que las diferentes unidades urbanas son las diferentes soluciones del problema.

El problema lo constituirá el tipo de espacio a analizar y todas aquellas características que permiten clasificarlo en una de las soluciones.

Si queremos solucionarlo mediante clasificación heurística, deberemos encajar cada uno de estos elementos en cada una de sus fases. Podemos comenzar identificando cada uno:

Problemas concretos

En este caso podemos considerar como problema concreto las características específicas que tiene el espacio como son:

- Las características concretas del barrio/distrito donde está ubicado (densidad de población, número de vías que pasan por él, densidad de tráfico)
- Distancias concretas a las unidades urbanas que ya existen
- Tipo de vías y densidad de trafico de las vías cercanas al espacio
- El tipo de espacio a clasificar

Problemas Abstractos

En este caso los problemas abstractos corresponderán a la generalización/abstracción de las características del entorno del espacio concreto.

Se podrían definir diferentes características que se derivarían de los criterios de clasificación que se indican en el enunciado. Los valores de estas características se obtendrían mediante reglas de abstracción cualitativa que usen criterios definidos por los expertos.

Las características que se indica en el enunciado son:

- Valoración cualitativa de las diferentes densidades de población (barrio/distrito)
- Valoración cualitativa de la densidad de tráfico (B/D)
- Valoración cualitativa de la proximidad a diferentes unidades urbanas (varias características, posiblemente una por cada tipo de unidad)
- Valoración cualitativa de la densidad de tráfico de la densidad de tráfico de las vías que están relacionadas con el espacio

La fase de abstracción de datos la realizarían las reglas que conviertan los datos de cuantitativos a cualitativos. Estas reglas serán mas o menos complejas dependiendo de la característica a abstraer.

Soluciones abstractas/soluciones concretas

En este caso se puede utilizar la clasificación de las unidades como dos niveles de solución, usando las unidades comerciales, unidades de servicio al ciudadano y unidades residenciales como soluciones abstractas. De todas maneras tal como esta detallado el enunciado es difícil hacer este nivel y requeriría alguna característica mas para poder separar primero estos tres tipos de soluciones. En el caso de tener estas características la fase de refinamiento nos llevaría a las tipologías concretas de unidades urbanas.

Una solución mas simple es clasificar directamente los espacios en las unidades urbanas específicas, utilizando estas como soluciones abstractas y evitando la fase de refinamiento que de hecho estaría incluida en la asociación heurística.

La asociación heurística hará la correspondencia entre lo problemas abstractos definidos por las características abstraídas y el tipo de espacio con los problemas concretos que serán las unidades urbanas. Estas reglas describirán todo el conocimiento que se explica en el enunciado.

Reglas ejemplo

En el caso de la clasificación de un colegio, en la primera fase habría que abstraer las características del espacio a analizar. Estas reglas de hecho serán comunes a todos los problemas.

Abstraeríamos por ejemplo los valores que corresponden a las densidades de población del Barrio/Distrito del espacio:

si Espacio.Barrio.Población > 300h/Km^2 entonces Densidad_Global = Media
si Espacio.Barrio.PoblaciónInfantil> 50h/Km^2 entonces Densidad_Infantil = Media

Igual para el trafico. En este caso los cálculos serán mas complejos:

si Espacio.Barrio.CalculaMediaVias> 1000coches/hora entonces Densidad_Trafico_Barrio = Alta

Abstraeriamos también características de cercanía a otras unidades:

si Espacio.ColegioMasCercano> 2Km entonces Cercanía_Colegios = Media
si Espacio.BibliotecaMasCercana< 200m entonces Cercanía_Bibliotecas = Al_lado

Algunas reglas podrían tener en cuenta el tipo de espacio para limitar los tipos de cercanías que se calculan.
Con estas reglas tendríamos ya todas las características abstractas que necesitamos. Ahora escribiríamos reglas que hicieran la asociación heurística.

si (Espacio.tipo=Manzana) y (Densidad_Infantil = Alta) y (Cercanía_Bibliotecas<=cerca) y (Cercanía_Zona_Verde <= Media) y (Densidad_Vías_Cercanas=Baja) y (Cercanía_Colegios={Media,Lejos})entonces Espacio.Unidad = Espacio.Unidad+{Colegio}

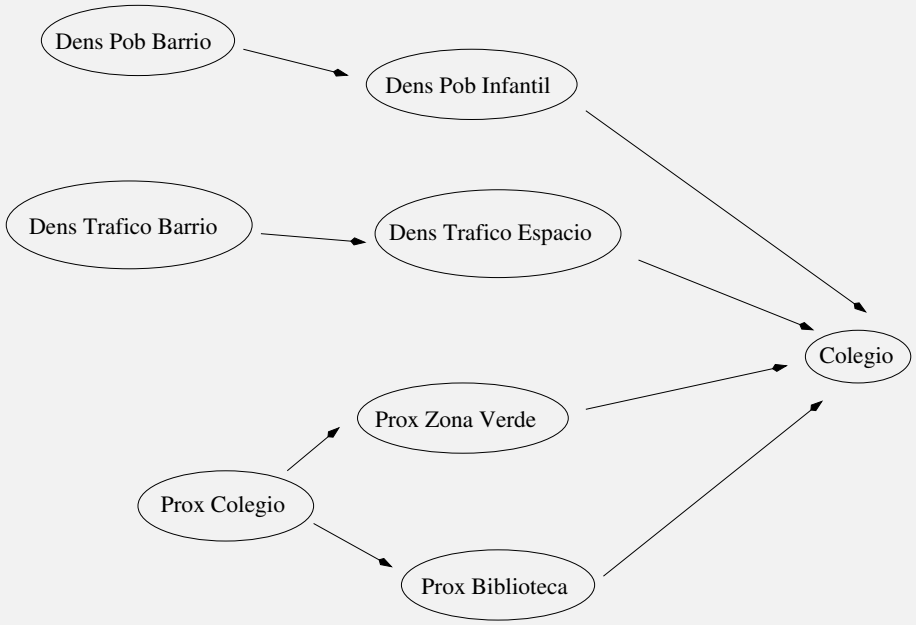
b) Una alternativa para representar el conocimiento de los ingenieros y las normas urbanísticas que permite obtener una propuesta de usos de los espacios vacantes son las redes bayesianas. Como ya se ha mencionado en el apartado anterior, en una manzana podría por ejemplo ubicarse una manzana de viviendas, un colegio, una plaza o un centro de asistencia primaria. Adicionalmente las evidencias recolectadas pueden hacer que alguno de los usos tenga más fuerza que otros. La red bayesiana permitiría representar el proceso de asignación de unidades y sus pesos.

Comenta cómo modelarías el problema con una red bayesiana (qué escogerías como nodos, qué valores tendrían y qué dependencias habría entre los nodos) y da un ejemplo simple de red bayesiana para ilustrarlo.

Lo que buscamos es describir la asignación de unidades a espacios usando una red bayesiana. Básicamente lo que hemos de hacer es replicar la fase de asociación heurística, por lo que tendríamos como nodos las características que abstraemos y las soluciones concretas.
Las soluciones concretas serían los nodos terminales en la red (lo que queremos predecir). Podemos representarlas de dos maneras:

- Un único nodo que contemplara como valores todas las posibles soluciones. Sería relativamente difícil el estimar las probabilidades ya que tendríamos que tener en cuenta muchas interrelaciones.
- Tantos nodos como soluciones, donde cada nodo tendría dos valores (si/no). A cada nodo terminal pueden asociarsele los padres que influyen en el de manera separada al resto.

El resto de nodos corresponderían a las características que hemos abstraído, podemos ponerlos como nodos independientes unos de otros o podemos descubrir las dependencias que existen entre ellas, por ejemplo entre las densidades de población o entre las densidades de tráfico del barrio y espacio o relaciones de proximidad entre unidades.
La principal ventaja que tiene esta aproximación es que no hace falta disponer de toda la información para dar un diagnóstico y ademas tenemos una valoración cuantitativa de la fuerza de cada posible solución.
Suponiendo que cada nodo terminal sea una solución podemos describir por ejemplo la red que clasifica un espacio como un colegio:



c) Determinar los posibles usos de cada uno de los espacios solo es el primer paso del proceso que sigue el departamento de planificación. El siguiente paso es encontrar la mejor forma de ocuparlos. En el caso particular de las unidades de interés ciudadano, se ha de tener en cuenta las unidades urbanas de ese tipo ya existentes, las diferentes restricciones de ubicación que impone el propio departamento y los recursos de que se disponen.

La ciudad determina que debe haber al menos un centro de atención primaria por barrio, pero no más de diez por distrito. Los centros de atención primaria han de estar lo más *lejos* posible entre si dentro del barrio. Debe haber al menos dos farmacias *cercanas* a cada centro de atención primaria, pero las farmacias del barrio deben estar a una distancia *media* entre ellas. Debe haber al menos un hospital por distrito y deben estar a distancia *media* de los centros de atención primaria.

Debe haber un colegio en un barrio por cada 300 niños, pero éste debe estar a una distancia *media* de otros colegios del barrio y debe estar *cerca* de al menos dos bibliotecas y al menos tres plazas en el barrio. Debe haber a distancia *media* del colegio al menos una zona verde y un polideportivo en el barrio.

El barrio debe tener al menos una zona verde y un polideportivo pero las áreas verdes y polideportivos de cada barrio deben estar a distancia *lejana* ente sí. Cada barrio debe tener al menos 5 plazas que deben estar a distancia *media* entre sí pero no más de 30 por distrito. No debe haber grandes vías *cerca* de colegios, zonas verdes o polideportivos.

Las oficinas de atención ciudadana deben estar *cerca* entre si en el barrio y no debe haber más de 5 por barrio. Debe haber como mínimo una biblioteca por barrio y debe estar *cerca* de alguna oficina de atención ciudadana.

Este problema se puede plantear como un problema de satisfacción de restricciones. Explica cómo se puede resolver con este enfoque. Identifica las variables adecuadas, sus dominios, los diferentes tipos de restricciones entre variables que se necesitan y la estructura de la red de restricciones. Justifica tus respuestas.

El plantear el problema como un PSR permitirá reducir las posibles asignaciones a espacio a solo las que son compatible con las normas que se imponen y a las otras restricciones adicionales, tanto locales como globales.

En este caso las variables serán los espacios, tanto los vacantes, como los ocupados, ya que la ocupación actual impone restricciones sobre qué podemos poner en los espacios vacantes.

Los valores serán para unas variables la ocupación actual y para otras las diferentes posibilidades de unidades de interés ciudadano en las que se haya clasificado el espacio. Obviamente solo usaremos las variables en las que entre sus posibilidades haya algún uso que sea unidad de interés ciudadano.

Habría que tener en cuenta que es posible que para un espacio puede que la restricciones no dejen ubicar ninguna de las posibilidades, por lo que debería haber un valor especial que indique no ocupación.

Respecto a las restricciones, según indica el enunciado habrá de diferentes tipos según los posibles valores que se pueden escoger. Podemos distinguir las que implican restricciones globales, que involucran todas o casi todas las variables (número máximo o mínimo de asignaciones de un tipo de valor por ejemplo). Por otro lado tenemos restricciones que relacionan una variables con sus vecinas y que también involucran varias variables (distancias, numero de unidades a una distancia, ...). De hecho las restricciones serán una función compleja que compruebe las condiciones que se explican en el enunciado.

La red se podría estructurar para simplificar la búsqueda utilizando los barrios, de manera que solo se tengan en cuenta durante la búsqueda donde sea necesario las variables que pertenecen al mismo barrio. Dentro de cada barrio las variables estarían conectadas todas con todas mediante relaciones n-arias que evaluarían las condiciones que representan las restricciones del problema.

5. La huelga de guionistas en EEUU ha abierto la puerta a los guiones generados por ordenador, asi que se nos ha planteado la posibilidad de diseñar un SBC capaz de generar el esquema del guión de episodios de series de televisión.

   Una serie tiene personajes, unos serán los protagonistas y otros los personajes secundarios. Cada personaje tendrá una serie de características, como por ejemplo una personalidad, una clase social, si es bueno o malo, ...

   Tendremos un conjunto de temáticas sobre las que podremos generar guiones como por ejemplo ciencia-ficción, acción, drama familiar, comedia de situación, ...

   El esquema del guión lo basaremos en secuencias ordenadas de estados por los que pasará cada personaje. Estos estados podrán ser emocionales (triste, alegre, enamorado, enfadado, ...) o físicos (peligro, hambre, herido, muerto, viaje). Esta secuencia de estados la determinaremos a partir del tipo de serie que queramos crear y las características de los personajes principales que intervienen.

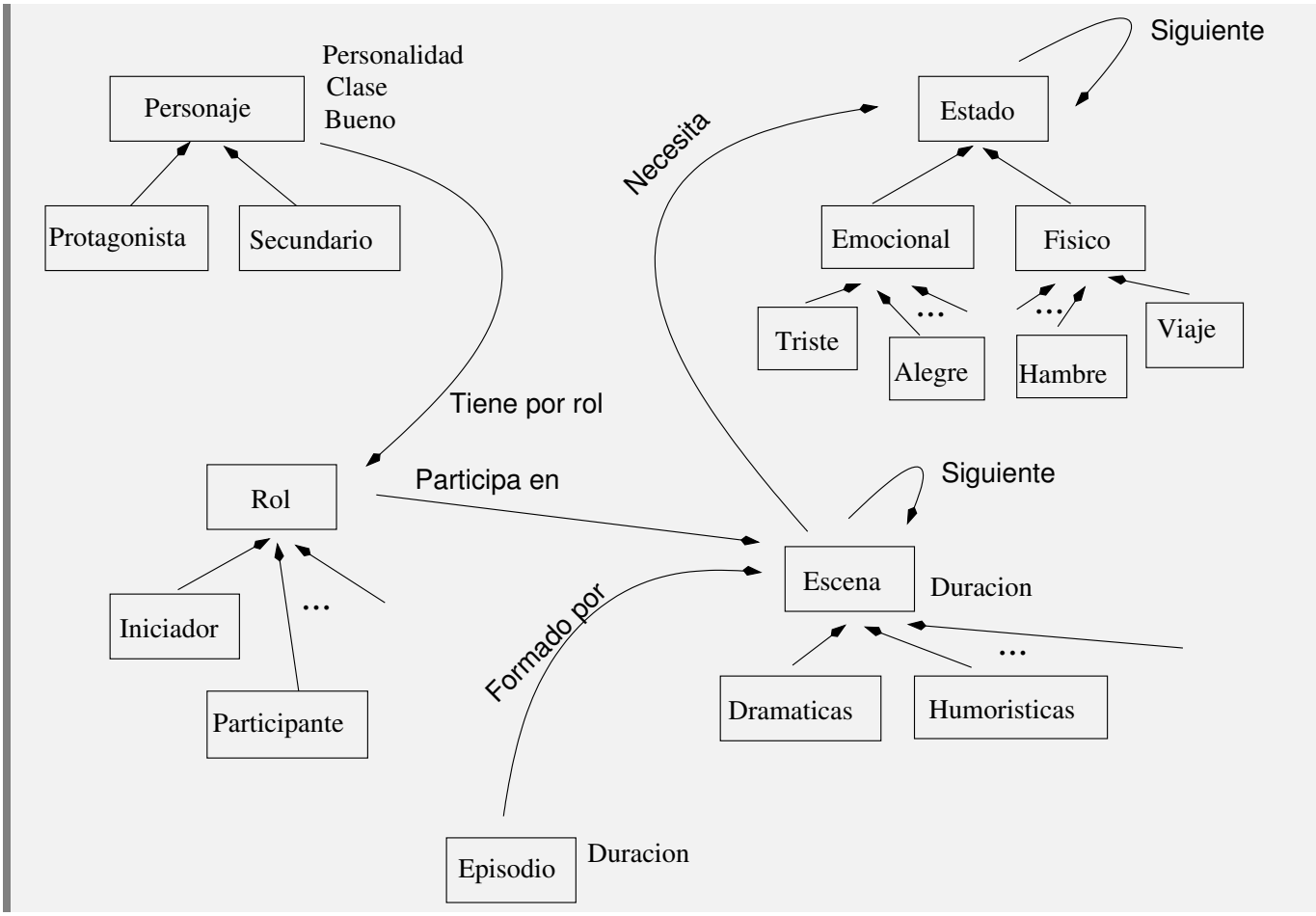
   El esquema propiamente dicho del guión será una secuencia ordenada de escenas. Estas escenas las clasificaremos en dramáticas (por ejemplo un rescate), humorísticas (un personaje le gasta una broma a otro), acción (una pelea), cotidianas (una familia desayunando), ... Una escena tendrá una duración asociada.

   Cada escena estará asociada al conjunto de estados necesarios para aplicarla. Una escena también tendrá asociados unos roles que indican qué personajes han de intervenir en la escena, los roles pueden ser por ejemplo iniciador, participante, receptor, ... Cada personaje estará asociado a una escena a través de un rol.

   El objetivo del SBC es crear el guión un episodio con una duración lo mas cercana posible a una dada, pero siempre inferior. Como datos de entrada nos indicarán qué personajes principales aparecen, cuales son sus características, la temática de la serie y detalles sobre la ambientación, como por ejemplo la época, el lugar, ... El episodio se compondrá de una serie de escenas y los personajes que intervienen en cada una de ellas. Todos los personajes principales tendrán que aparecer en algún momento y tendremos que determinar qué personajes secundarios hacen falta, con la restricción de que deberemos añadir el mínimo posible ya que los actores son caros.

   a) Identifica todos los conceptos que forman parte del problema. Representa gráficamente estos conceptos mediante una red de frames incluyendo los atributos mas relevantes y las relaciones tanto taxonómicas como no taxonómicas que creas que son necesarias.

Estos son los conceptos principales que permiten describir los elementos del problema y su solución. Hay una serie de conceptos que no se han incluido en la ontología dado que solo corresponden a datos de entrada con los que se elaborará la solución pero que no pertenecen directamente a ninguno de los conceptos principales o la solución, como por ejemplo la temática de la serie o la ambientación. No obstante podría añadir el un concepto *serie* e incluirlos ahí.



b) Hemos decidido que el primer subproblema a resolver es el determinar los estados por los que pasará cada personaje. Para hacer mas sencillo el problema hemos escogido un conjunto finito de secuencias de estados típicas y una serie de características propias que las describen. Lo que queremos es asignar una de ellas a cada personaje principal dependiendo de la información de entrada del problema. Estas secuencias se podría refinar con algunas características específicas de los personajes o de la serie.

Por ejemplo, podríamos definir la secuencia *torbellino emocional genérico* que correspondería a la secuencia de estados (triste, enamorado, alegre, engañado, (triste o vengativo)) que podríamos describir por las características (soporta estrés = si, fortaleza emocional = si, fortaleza física = indefinida, aspecto físico = normal), donde se podía definir la elección sobre el último estado dependiendo por ejemplo de si el personaje es bueno o malo.

Este planteamiento encaja con el método de clasificación heurística. Explica lo que se debería hacer en cada una de las fases de esta metodología para resolver el problema tal como se ha explicado. Pon ejemplos sencillos de reglas para cada fase.

Para plantear un problema mediante clasificación heurística debemos definir tres fases:

- Abstracción de datos
- Asociación heurística
- Refinamiento de la solución

A) Abstracción de datos

Dado que hemos definido cada una de las secuencias de acciones a partir de una serie de características solo tendremos que abstraer las características de los personajes y de la serie de estas.

Definiremos la fase de abstracción de datos como un conjunto de reglas que obtienen los valores que corresponden a las características sobre las que se definen las secuencias, por ejemplo:

serie.género=comedia ∧ personaje.bueno=no ⇒ fortaleza física=no
serie.género=acción ∧ personaje.bueno=si ⇒ fortaleza física=si
personaje.personalidad=alegre ∧ personaje.género=femenino ⇒ fortaleza emocional=si

B) Asociación heurística

La fase de asociación heurística deberá seleccionar la secuencia de acciones más adecuada para las características abstraidas. Como hemos definido cada secuencia por unas características solo tendremos que escribir las reglas que hagan la asociación, por ejemplo:

fortaleza física=indefinida ∧ aspecto físico=normal ∧ soprta estrés=si ∧ fortaleza emocional=si ⇒ secuencia=torbellino emocional genérico

C) Refinamiento de la solución

En la fase de refinamiento de la solución se adaptarán los detalles variables de la secuencia a las características específicas del personaje.
secuencia=torbellino emocional genérico ∧ personaje.bueno=no ⇒ secuencia=torbellino emocional final vengativo

c) El segundo subproblema consiste en construir la secuencia de escenas que encajen lo mejor posible con los estados por los que han de pasar los personajes principales (los estados que no se puedan encajar deberían corresponder a personajes secundarios) y determinar los personajes necesarios para cada escena. Éste es un problema de síntesis que podríamos resolver mediante proponer y aplicar. Define un conjunto de operadores que permitan resolver el problema, indicando que harían, las restricciones globales y específicas que deberían tenerse en cuenta para aplicarse y los criterios de evaluación que determinan la bondad de cada operador.

En el método de proponer y aplicar debemos tener un conjunto de operadores de entre los que escoger que nos vayan construyendo la solución. Los operadores se escogerán respecto a criterios globales y específicos sobre la solución en curso.

En este caso tenemos que encajar un conjunto de escenas sobre un conjunto de secuencias de estados y asignar los personajes a los roles de las escenas.

Las restricciones globales son el que no superemos la duración del episodio, hagamos aparecer a todos los personajes principales y que minimicemos el número de personajes secundarios.

Como operadores podriamos tener:

- añadir/quitar/intercambiar escenas
  - Estos operadores modificarían la secuencia de escenas de la solución.
  - Se evaluarían respecto a como de bien encajan con los estados que hay en la solución (cuantos más estados de la escena ya existan en la solución mejor) y el número de personajes necesarios en la escena (cuantos menos mejor, ya que no queremos muchos personajes secundarios).
- Determinar el rol de una escena
  - Escogería el personaje que corresponde al rol de una escena.
  - Se evaluaría respecto a si permite asociar el rol a personajes principales que no han aparecido o si hace aparecer personajes secundarios nuevos.

6. Una compañía aérea desea organizar las asignaciones de puestos que han de desempeñar sus empleados de manera que se puedan cubrir todos los puestos que necesitan para atender los vuelos de la compañía que llegan y salen de un aeropuerto concreto.

   El personal de que dispone la compañía se puede clasificar en personal de vuelo y personal de tierra. Para el personal de vuelo hay puestos de piloto y auxiliar de vuelo. Para el personal de tierra hay puestos en facturación de equipajes, venta de billetes, atención al cliente, embarque y reclamación de equipajes. Cualquier persona del personal de tierra puede ocupar cualquiera de los puestos posibles. Por razones evidentes no ocurre lo mismo con el personal de vuelo. Para el personal de tierra cada

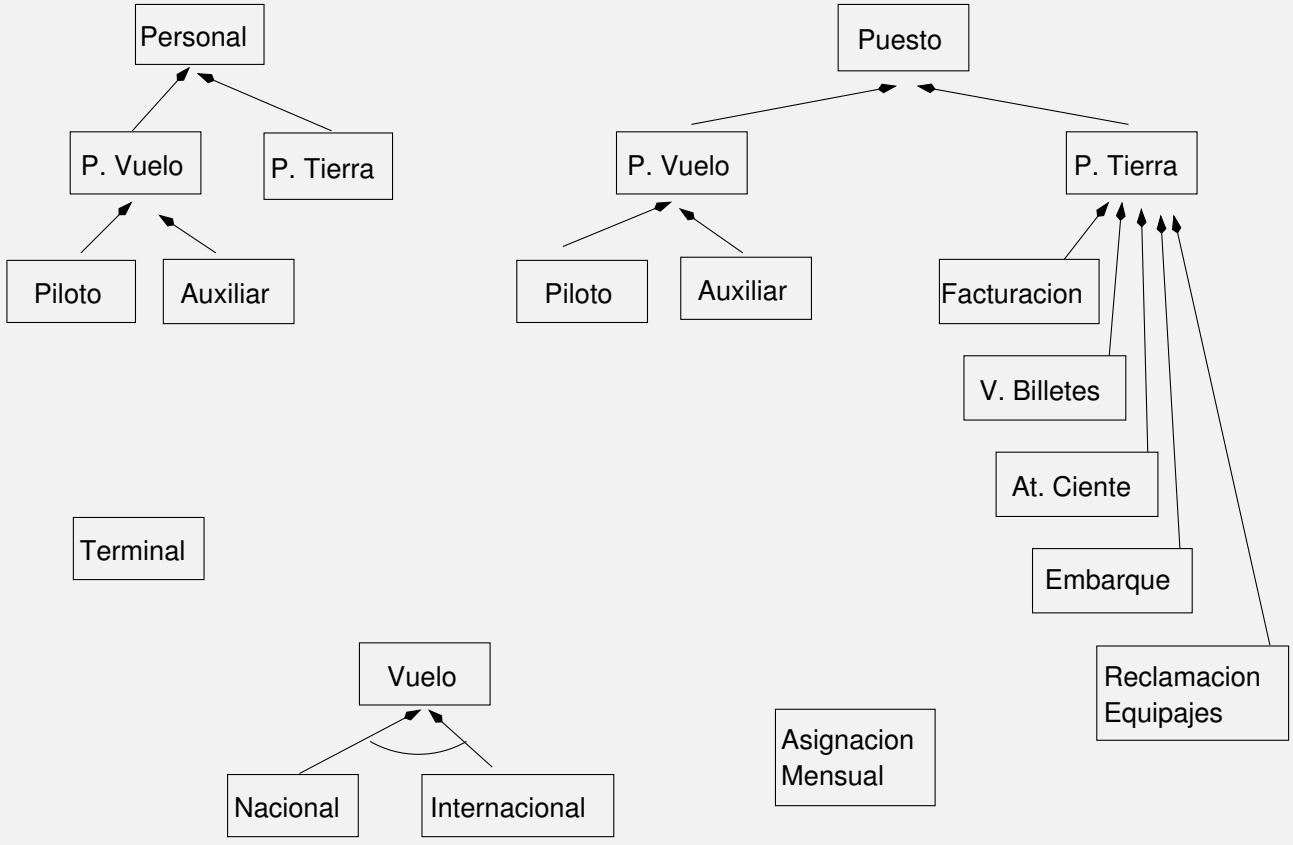
puesto se asigna a una persona durante una semana en cualquiera de los tres horarios de trabajo (mañana, tarde y noche). El personal de vuelo es móvil, ya que se desplaza en los vuelos y por lo tanto sólo puede ser asignado a un nuevo vuelo cuando ha regresado del ultimo. Los puestos de tierra están asignados a las diferentes terminales del aeropuerto. En cada terminal hay facturación de equipajes, embarque y atención al cliente, pero solo hay venta de billetes y reclamación de equipajes en una terminal.

Para poder hacer la asignación de personal nos hará falta cierta información para cada persona como por ejemplo cual fue el último puesto que ocupó, cuándo lo hizo, en qué horario, etc.

La compañía ha de cubrir un número específico de puestos, tanto para vuelos como en tierra. Esta asignación deberá cumplir también ciertas restricciones, como por ejemplo que el personal de tierra no pase más de dos semanas en el mismo puesto, salvo si está en reclamación de equipajes, puesto en el que nadie puede estar más de una semana seguida en un mes. Tampoco puede pasar una persona más de un mes en el mismo horario. El personal de vuelo no puede realizar más de un vuelo internacional al mes y no debe ser asignado a más de 7 vuelos al mes.

El objetivo es construir un sistema capaz de obtener una asignación de personas a puestos en el aeropuerto durante un mes para cada uno de los horarios cumpliendo las restricciones comentadas.

a) Identifica qué conceptos, características, objetivos y soluciones forman el problema. Representa gráficamente los conceptos que has identificado mediante una red de frames y sus relaciones taxonómicas. Incluye en cada concepto al menos dos atributos que sean necesarios para su descripción. Describe completamente mediante el lenguaje de representación de frames vista en clase al menos tres relaciones entre los conceptos que has representado. Indica qué demons podrían ser necesarios en la representación y qué comprobarían (no hace falta que los implementes).



En las asignaciones de puestos es necesario conocer la historia anterior, por lo que se debería conocer la fecha en la que se ha asignado un puesto.

Atributos:

- Para personal: Nombre, DNI, ...
- Para puesto: Horario, fecha inicio, fecha fin, ...
- Para Terminal: Identificador, Aeropuerto, ...
- Para Vuelo: Fecha salida, destino, origen, ...
- Para asignación mensual: Mes, numero de personas asignadas, ...

Relaciones:

Asignado a, entre personal y puesto.

Relación	Asignado a
Dominio	Personal
Rango	Puesto
Cardinalidad	N
Inversa	Ocupado por (1)
Transitiva	no
Compuesta	no
Demons	—
Herencia	—

Suponemos que tenemos diferentes instancias para un puesto en diferentes fechas, de manera que un puesto solo puede estar ocupado por una persona, pero una persona tendrá muchos puestos a lo largo del tiempo.

Desempeñado en, entre puesto de tierra y terminales

Relación	Desempeñado en
Dominio	Puesto tierra
Rango	terminal
Cardinalidad	1
Inversa	Asignados (N)
Transitiva	no
Compuesta	no
Demons	—
Herencia	—

Embarcado en, entre puesto de vuelo y vuelo

Relación	Embarcado en
Dominio	Puesto vuelo
Rango	vuelo
Cardinalidad	1
Inversa	tripulacion (n)
Transitiva	no
Compuesta	no
Demons	—
Herencia	—

Demons:

En las relaciones que asignan puestos a personal podemos tener un emos que compruebe que el personal de vuelo no puede ocupar puestos del personal de tierra y viceversa, también que un auxiliar de vuelo no puede hacer de piloto y viceversa.

También podemos comprobar restricciones del problema mediante los demosn, por ejemplo si asignamos una persona de la categoría de personal de vuelo a un puesto que corresponde a un vuelo internacional se puede comprobar que no haya sido asignado a otro vuelo internacional en ese mes. Se podría hacer lo mismo con otras restricciones.

La otra alternativa sería hacer esas comprobaciones mediante reglas durante el proceso de resolución del problema que es lo que se planteará en el último apartado.

b) El problema que se pretende resolver ¿es un problema de análisis o de síntesis? ¿Porqué?

Dado que lo que pretendemos es construir una solución a partir de loas elementos que tenemos (personal, puestos) lo que resolvemos es un problema de síntesis.

c) ¿Si lo implementáramos mediante un sistema de producción que tipo de razonamiento te parecería el más adecuado para resolverlo? ¿Porqué?

Dado que es un problema de síntesis solo tiene sentido el razonamiento hacia adelante, ya que para aplicar el razonamiento hacia atrás deberiamos disponer de una solución de la que partir.

d) ¿Como descompondrías la solución del problema? Identifica y especifica los subproblemas a resolver y como se encadenarían para construir la solución. Indica si los subproblemas se corresponden con alguna de las metodologías de resolución de problemas que conoces. Escribe para cada problema que identifiques algún ejemplo de las reglas de producción que harían falta para resolverlo.

Se pueden distinguir dos problemas independientes, asignar el personal de tierra y asignar el personal de vuelo.

Como no hay interacción entre ellos, se pueden resolver por separado.

Cada problema se podría dividir en dos fases, una primera que buscara todas aquellas personas que pueden asignarse a puestos porque no violen las restricciones del problema y una segunda fase en la que se buscará en el espacio de posibles asignaciones.

El método más adecuado de resolución de problema de los vistos en teoría es la **resolución constructiva**. Evidentemente la solución se va construyendo durante el proceso de resolución.

Una posibilidad sería aplicar el método *proponer y aplicar*. Se podrían realizar primero las fases de selección de candidatos dejando la fase de asignación al método de proponer y aplicar para elaborar la solución siguiendo los pasos que propone la metodología.

Primero se obtendrían todas las asignaciones que son posibles para cada persona evaluando mediante un conjunto de reglas la adecuación de cada una de ellos. Estas reglas evaluarían las diferentes restricciones que impone el problema para poder ocupar un puesto teniendo en cuenta las asignaciones previas.

A partir de esas asignaciones de candidatos se utilizarían operadores de asignar/desasignar/mover una persona a uno de los puestos que hay libres durante el mes en el que se tiene que elaborar la solución sin violar ninguna restricción. Se evaluaría la bondad de cada asignación respecto al respeto de las restricciones, se reevaluarían las asignaciones candidatas que se tuvieran eliminando las que no fueran posibles y se evaluaría la posibilidad de poder completar la solución. Se elegiría la asignación que permitiera continuar la solución.

De hecho este problema replicaría la solución que se podría obener mediante un algoritmo de satisfacción de restricciones.

Ejemplos de reglas:

si todas las asignaciones que tiene un piloto no son internacionales entonces candidato a vuelo internacional

si una persona esta asignada a reclamación de equipajes entonces no es candidato a otra asignación en reclamación de equipajes

si un piloto tiene siete asignaciones de vuelo entonces no es candidato a otra asignación de piloto

8. El restaurant *Aprofita-ho* és un restaurant de prestigi internacional. S'ha guanyat el seu prestigi perquè elabora els seus plats tenint en compte les preferències del client i els ingredients dels que disposa. Així doncs, els menús resultants són al gust del client. el xef té en compte si el client prefereix la carn crua o cuita, si li a grada el peix, si és vegetarià, si li agraden les verdures poc cuites, si li agraden les amanides, si prefereix les amanides a l'inici o al final, si li agrada la barreja entre dolç i salat, etc. També té en compte els ingredients dels que disposa: ous, pollastre, conill, enciam, endívies, col, patates, llobarro, rap, calamars, tomàquet, ceba, alls, etc. Amb tota aquesta informació, elabora els plats i confecciona el menú mès adient per al client.

El xef pertany a la nova fornada de cuiners tecnócrates, i està força convençut de que es pot informatitzar el procés d'elaboració dels plats i confecció del menú. Encara més, creu que podría ser útil un sistema expert, segons el que va apendre en un curs de postgrau en Intel·ligencia Artificial que va fer a la UPC. Responeu a les següents qüestions:

a) Identifica qué conceptes forman les dades d'entrada del problema. Representa gràficament aquests conceptes mitjançant una xarxa de frames inclouent els atributs mes relevants i les relacions, tant les taxonómiques como les no taxonómiques que creguis que son necessaries.

Podemos distinguir tres elementos básicos (conceptos): Preferencias, ingredientes y platos. Adicionalmente los platos se combinan en menús.

Respecto a las preferencias, del texto se deduce que pueden ser sobre la elaboración del plato (carne poco hecha), sobre el topo genérico del plato (carne, pescado, verdura) o sobre el orden de los platos (primer ensaladas, ...).

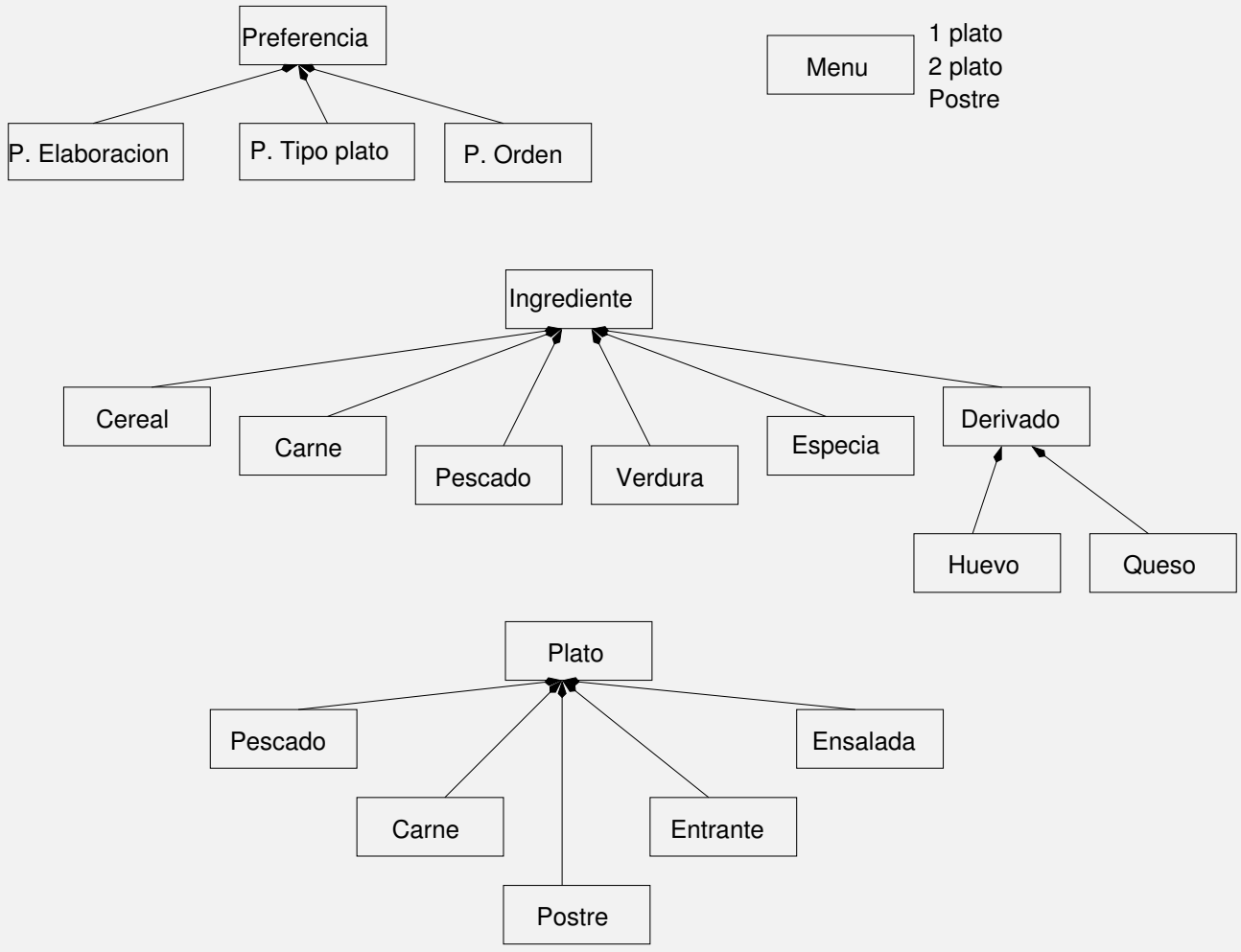
Para los ingredientes, podemos estructurarlos como mejor nos parezca ya que no hay indicaciones.

Los platos deben estructurarse de manera que sea sencillo acceder a ellos según las preferencias que tiene el usuario y los ingredientes que los componen.

Deberiamos caracterizar cada plato indicando propiedades que después permitan hallar la solución, por ejemplo:

- Si es primer o segundo plato, o ambos, o postre.
- Si el plato es pesado o no.
- Si es dificil de elaborar.
- Si es un plato que agrada a la mayoría de la gente.
- ...
- Añadiendo la relacion que indica si combina un plato combina bien con otros

Podríamos hacer la siguiente representación de estos conceptos:



El concepto menú forma parte de la solución.

b) Identifica els conceptes que forman la solució del problema. Representa gráficament aquests conceptes mitjançant una xarxa de frames i les relacions, tant les taxonómiques, como les no taxonómiques, que creguis que son necessaries.

En este caso la solución estará compuesta por el concepto menú que estará relacionado con los platos que se hayan elegido. A la representación del apartado anterior hay que añadirle una relación que indique que una instancia de plato está relacionado con una instancia de menú.

c) ¿Com descompondries el problema en subproblemes? Identifica i especifica els subproblemes a resoldre i com s'encadenarien per construir la solució.

Podemos identificar tres subproblemas:

1) Identificación de los gustos del cliente.
2) Selección del conjunto de platos que podría formar parte de la solución (esto incluye usar los gustos del cliente y las restricciones de ingredientes).
3) Combinar los platos del menú (2 platos + postre), maximizando las preferencias.

El primer subproblema se podría simplificar a escoger un tipo de usuario de entre un conjunto de usuarios preestablecido (vegetariano, carnívoro, fan de las ensaladas, mixto vegeta-carnívoro, ...). Otra opción sería tener en consideración cada posible restricción que se puediera obtener del usuario y tener una identificación a menor nivel de granularidad. La ventaja de obtener un perfil es poder dirigir la resolución en el resto de subproblemas, disponer de un perfil puede permitir seleccionar directamente platos y menús acorde con esos perfiles con las restricciones que les podamos asignar focalizando más la búsqueda.

Este subproblema estaría implementado mediante un conjunto de reglas que preguntaran al usuario sus preferencias siguiendo un árbol de preguntas que guiara hacia su identificación de la manera mas eficiente posible. Se podrían organizar las reglas primero preguntando preferencias generales (vegetariano, ...) y despues preguntas másа específicas (preferencia o no de ingredientes concretos, ...)

De este subproblema obtendríamos un conjunto de restricciones y preferencias. Si decidieramos identificar usuarios tipo tendríamos como conclusión alguno del os usuarios que definieramos y eso nos daría un conjunto de preferencias y restricciones específicas para ese tipo.

El segundo problema tendría en cuenta las restricciones del usuario (las que se refieren directamente a los platos) y las de ingredientes, para eliminar del espacio de búsqueda aquellos platos que no se pueden elaborar.

Este subproblema solo tendría que evaluar cada plato que se puede elaborar y asignarlo o no al conjunto de platos con los que elaboraremos los menús.

De este subproblema obtendríamos como conclusiones un conjunto de platos.

El tercer problema requiere construir el menú a partir de los platos disponibles teniendo en cuenta otras restricciones del usuario (primero ensalada, mecla dulce y salado, carne con pescado, ...) y otras de sentido común (no dos platos de pescadao, no dos platos con la misma guarnición, ...) para elegir la combiandión de platos adecuada que maximice las preferencias.

La implementación de este subproblema podría tener un conjunto de reglas que puntuaran los diferentes platos según las preferencias del usuario y sus posibilidades de combinar y otro conjunto de reglas que generaran las combinaciones válidas. El sistema podría escoger la combinación que tuviera la mejor puntuación. Esto podría asimilarse a una búsqueda en el espacio de meús posibles donde unas reglas harían la construcción de la solucion y otras harían la evaluación. En este caso la estrategia de búsqueda elaboraría múltiples soluciones a la vez quedándose con la mejor.

De este subproblema obtendríamos el menú que mas se adapta a las preferencias y restricciones del usuario.

d) Indica per cada subproblema que has identificat si es un problema d'anàlisis o de síntesi. Justifica-

ho.

El primer problema es de análisis si elaboramos un perfil escogiendo de entre un conjunto de diferentes tipos de usuarios. Si solo recogemos las restricciones del usuario no se puede asignar a ninguna de las dos categorías.

El segundo problema tampoco se puede asignar a ninguna de las dos categorías ya que solamente nos limitamos a propagar las restricciones en el conjunto de platos para obtener los elementos del espacio de búsqueda del tercer problema.

El tercer problema es de síntesis ya que construimos una solución a partir de un conjunto de elementos.

e) ¿Quin tipus de raonament et sembla mes adequat para cada subproblema? Justifica-ho.

Todos los problemas se pueden resolver mediante razonamiento hacia adelante ya que no disponemos de soluciones de las que partit. El primer subproblema si se plantea como un problema de análisis se podría resolver mediante razonamiento hacia atrás si el número de perfiles no es grande.

11. La consultoria informàtica VADEMIRACLE S.A. ha rebut l'encàrrec de la Conselleria d'Indústria de dissenyar un sistema informàtic intel· ligent que permeti facilitar la selecció i posterior compra d'automóbils nous per als ciutadans, dins la campanya de foment de la renovació del parc automobilístic i del consum. Aquest sistema d'ajut a la presa de decisions s'instalarà en diversos locals de la Generalitat arreu del país. El sistema ha de tenir en compte les necessitats, gustos, pressupost, etc., de les persones, i recomanar-els-hi el/s cotxe/s que millor s'adequen a les seves demandes. El/s cotxe/s recomenats seran models determinats de les diferents marques automobilístiques disponibles, amb totes les seves característiques ben determinades (poténcia, nombre de portes, tipus d'equipament, etc.) i havent tingut en compte els equipaments opcionals desitjats. Responeu a les següents qüestions [4 punts]:

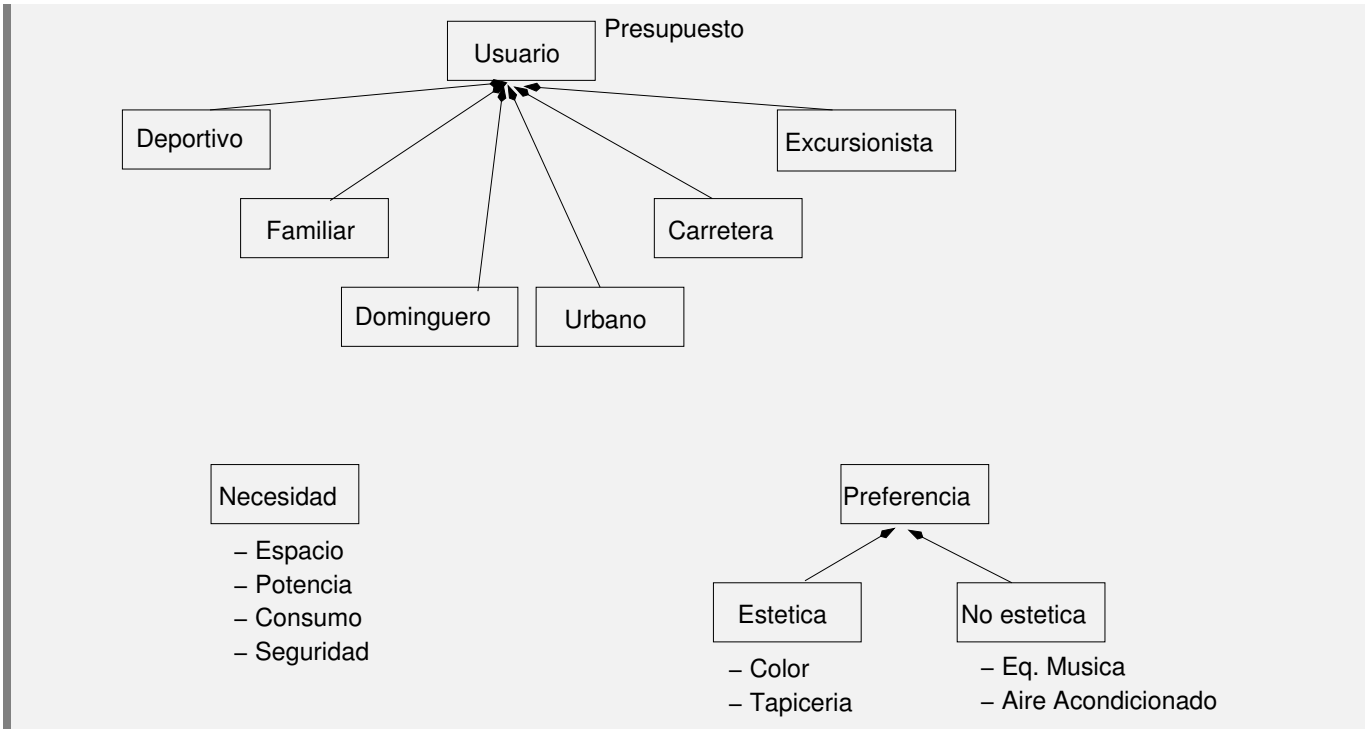
a) Identifica qué conceptes forman les dades d'entrada del problema. Representa gràficament aquests conceptes mitjançant una xarxa de frames inclouent els atributs mes relevants i les relacions, tant les taxonómiques como les no taxonómiques que creguis que son necessaries.

Los datos de entrada pertenecen a tres conceptos: Usuarios, necesidades y preferencias.

Podríamos representar una relación entre usuarios y necesidades y preferencias, pero dado que el sistema solo trata un usuario a la vez la relación puede quedar implícita.
De cara a la resolución del problema el concepto usuario se puede especializar en un conjunto de usuarios tipo que permitan después hacer la recomendación. Los atributos de un usuario podrían incluir características relevantes para la recomendación, como si tiene hijos, es soltero, le gusta ir de fin de semana, hace escapadas al campo, ...

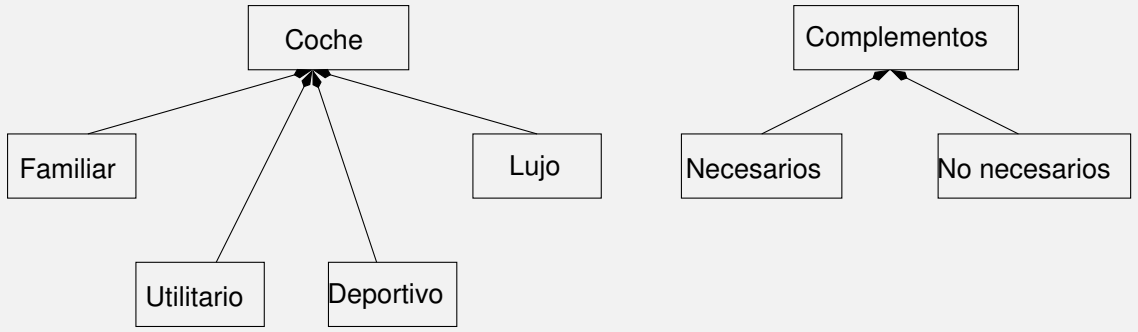
Las necesidades se pueden particularizar en diferentes conceptos como espacio, potencia, seguridad, o se pueden considerar estos conceptos como atributos del concepto necesidad. Las preferencias se pueden especializar en estéticas y no estéticas y se referiran a los elementos adicionales que pueda tener el coche, como el color, el tipo de tapicería, o la inclusión de un equipo de música o aire acondicionado. Estas especializaciones podían tener los elementos correspondientes como nuevas especializaciones o considerarse también atributos.

La representación de estos conceptos sería la siguiente:



b) Identifica els conceptes que forman la solució del problema. Representa gráficament aquests conceptes mitjançant una xarxa de frames i les relacions, tant les taxonómiques, como les no taxonómiquess, que creguis que son necessaries.

Podemos incluir dos conceptos en la solución, por un lado el concepto coche, que podría estar especializado en tipos de coche según sus características pudiendo tener diferentes niveles de especialización. Sus atributos se corresponderán con las necesidades y las preferencias. Como un concepto adicional podemos incluir el equipamiento (aunque podría formar parte del concepto coche) y lo podríamos especializar en necesario y no necesario. Estaría ligado también a las preferencias del usuario.



c) ¿Com descompondries el problema en subproblemes? Identifica i especifica els subproblemes a resoldre i com s'encadenarien per construir la solució.

Hay dos maneras de abordar el problema, o podemos utilizar un conjunto de perfiles de usuario para focalizar el proceso de resolución o intentamos directamente hacer cuadrar las necesidades y preferencias del usuario con los coches disponibles.

La aproximación que permite estructurar mejor la resolución es la primera.

Con esta aproximación podemos identificar los subproblemas con las diferentes fases de la clasificación heurística ya que se trata de un problema de análisis. HEmos de abstraer las necesidades y gustos del usuario de manera que lo clasifiquemos en una de sus especializaciones y a través de ellas dirijamos la solución al coche más adecuado. Finalmente se puede refinar la solución escogiendo unas características concretas.

A) Abstracción de datos

Recopilariamos la información sobre las caracterísitcas, necesidades y preferencias del usuario y las identificariamos con alguno de sus especializaciones. Tendriamos reglas como por ejemplo:

si necesidad de espacio grande y famila numerosa entonces usuario familiar
si soltero y juerguista y potencia alta entonces usuario deportivo
si presupuesto bajo y solo fin de semana entonces usuario dominguero

B) Asociación heurística

Asociamos los usuarios tipo a soluciones abstractas usando características adicionales sobre los coches. Tendriamos reglas como por ejemplo:

si usuario dominguero y seguridad alta entonces coches pequeños y seguros
si usuario deportivo y presupuesto alto entonces deportivos de gama alta
si usuario familiar y presupuesto alto entonces monovolumen de gama media

C) Fase de refinamiento

Una vez determinada la categoría de coche deberíamos concretar una marca y modelo específicos y escoger los complementos adecuados a las características del usuario. Tendriamos reglas como por ejemplo:

si monovolumen y num personas > 7 entonces monovolumen con mas de 7 plazas
si deportivo de gama alta y gusta musica entonces complemento CD
si deportivo de gama alta y color rojo entonces ferrari testarrosa

d) Indica per cada subproblema que has identificat si es un problema d'anàlisis o de síntesi. Justifica-ho.

Como se ha comentado el problema corresponde a un problema de análisis y los subproblemas se pueden encajar en la metodología de clasificación heurística.

e) ¿Quin tipus de raonament et sembla mes adequat para cada subproblema? Justifica-ho.

Dado que tenemos un conjunto de soluciones fijas y no muy grande para los usuarios se podría utilizar tanto el razonamiento hacia adelanta como el hacia atrás para la primera fase, en la segunda podríamos tambien escoger las dos alternativas, aunque el número de tipos de coches será mayor que el de usuarios. Para la parte de refinamiento al tener mas posibilidades lo mas adecuado sería aplicar razonamiento hacia adelante.

6.1 Problemas solucionados

2. Las siguientes oraciones corresponden a la primera oración (detectadas por el primer punto y levemente retocadas para eliminar caractéres extraños y agrupar los nombres de persona) de varias páginas de la wikipedia en castellano correspondientes a personas:

   1) Pablo_Ruiz_Picasso (n. Málaga, España, 25 de octubre de 1881 - Mougins, Francia, 8 de abril de 1973), conocido como Pablo_Picasso, fue un pintor y escultor español, creador, junto con Georges_Braque y Juan_Gris, del movimiento cubista.
   2) Paul_Auster (Newark, Nueva_Jersey, 3 de febrero de 1947) es un escritor estadounidense, Premio Príncipe de Asturias de las Letras 2006.
   3) Pierre_Corneille (Ruán, Francia, 6 de junio de 1606-París, 1 de octubre de 1684), dramaturgo francés.
   4) Juan_Marsé_Carbó (Barcelona, 8 de enero de 1933), es un destacado novelista español de la llamada Generación de los 50, concretamente de la denominada Escuela de Barcelona, de la que también formaron parte sus amigos Jaime_Gil_de_Biedma, Carlos_Barral, Juan_García_Hortelano, Manuel_Vázquez_Montalbán, Juan_Goytisolo, Terenci_Moix y Eduardo_Mendoza.
   5) Jaume_Cabré_Fabré (Barcelona, 1947) es un filólogo y escritor catalán.
   6) Antoni_Tàpies_i_Puig, marqués de Tàpies (Barcelona, 13 de diciembre de 1923) es un pintor, escultor y teórico del arte español.
   7) Antonio_López_García (Tomelloso, Ciudad_Real, 6 de enero de 1936) es un pintor y escultor español.

   Consideramos que la wikipedia puede ser una fuente excelente de información y, concretamente, que el análisis de la primera oración de las páginas de la wikipedia que corresponden a personas nos puede proporcionar:

   - el nombre de la persona
   - la fecha de nacimiento
   - el lugar de nacimiento
   - la fecha de defunción
   - el lugar de defunción
   - la actividad de la persona
   - la nacionalidad

   Tened en cuenta que además de esta información, las oraciones contienen material irrelevante para nuestros intereses (por ejemplo, que Tàpies sea marqués o que Marsé fuera amigo de Vázquez Montalbán).

   Considerad ahora la siguiente DCG y su lexicón correspondiente:

```
oracion --> quien, que, predicado, punto.
quien --> [Q], {esPersona(Q)}.
que --> leftB, lugar, comaOpcional, fecha, rightB.
predicado --> vc, sn.
lugar --> [X],{esCiudad(X)}.
lugar --> [X],{esCiudad(X)}, comaOpcional, [Y],{esProvincia(Y)}.
fecha --> [X],{esAny(X)}.
fecha --> [X],{esDia(X)}, separador, [Y],{esMes(Y)}, separador, [Z],{esAny(Z)}.
sn --> det, ns.
sn --> det, ns, adj.
```

```
ns --> n.
ns --> n1,conj,n.
n1 --> n.
n1 --> n,coma,n1.
comaOpcional --> [].
comaOpcional --> coma.
separador --> [].
separador --> [de].
vc --> [es].
coma --> [','].
punto --> ['.'].
leftB --> ['('].
rightB --> [')'].
det --> [un].
conj --> [y].
n --> [N], {esProfesion(N)}.
adj --> [A], {esNacionalidad(A)}.

esProfesion('pintor').
esProfesion('escultor').
esProfesion('escritor').
esProfesion('novelista').
esProfesion('filólogo').
esProfesion('dramaturgo').
esNacionalidad('catalán').
esNacionalidad('español').
esNacionalidad('francés').
esDia(D):-   number(D),D > 0, D < 32.
esMes(M):-   number(M),M > 0, M < 13.
esMes('abril').
esMes('diciembre').
esMes('enero').
esMes('febrero').
esMes('junio').
esMes('octubre').
esAny(D):-  number(D).
esPersona('Pablo_Ruiz_Picasso').
esPersona('Georges_Braque').
esPersona('Pablo_Picasso').
esPersona('Juan_Gris').
esPersona('Paul_Auster').
esPersona('Pierre_Corneille').
esPersona('Juan_Marsé_Carbó').
esPersona('Jaime_Gil_de_Biedma').
esPersona('Carlos_Barral').
esPersona('Juan_García_Hortelano').
esPersona('Juan_Goytisolo').
esPersona('Manuel_Vázquez_Montalbán').
esPersona('Terenci_Moix').
esPersona('Eduardo_Mendoza').
esPersona('Jaume_Cabré_Fabré').
esPersona('Antoni_Tàpies_i_Puig').
esPersona('Antonio_López_García').
```

```
esPais('España').
esPais('Francia').
esProvincia('Nueva_Jersey').
esProvincia('Ciudad_Real').
esCiudad('Málaga').
esCiudad('Mougins').
esCiudad('Newark').
esCiudad('Ruán').
esCiudad('Tomelloso').
esCiudad('Barcelona').
```

Se pide :

a) Decid qué oraciones analizaría correctamente la DCG.

Es fácil ver que sólo las oraciones 5 y 7 son analizadas correctamente

b) Qué modificaciones o adiciones deberían incorporarse a la gramática y lexicón para que la DCG funcionara correctamente sobre las 7 oraciones (y otras similares).

Un problema es la presencia de material no relevante en varias posiciones, podemos consumirlo con:

```
noRelevante -->[].
noRelevante -->[_], noRelevante.
```

y colocar la llamada a noRelevante donde convenga:

```
oracion --> quien, que, predicado, noRelevante, punto.
```

Nos soluciona la oración 2, siempre y cuando se añada estadounidense como nacionalidad en el lexico.

La presencia de comas en algunos puntos (por ejemplo, "Juan_Marsé_Carbó (Barcelona, 8 de enero de 1933), ..." puede solucionarse incluyéndola como opcional:

```
oracion --> quien, que, comaOpcional, predicado, noRelevante, punto.
```

También aparecen adjetivos (por ejemplo, "destacado") acompañando al nombre. Podemos incorporarlos como opcionales y a ambos lados del nombre:

```
adj(m) --> [destacado].
...
adjOpcional --> [].
adjOpcional --> adj.
ns --> adjOpcional, n, adjOpcional.
n1 --> adjOpcional, n, adjOpcional.
```

Con ello ya tenemos analizadas correctamente las oraciones 2, 4, 5 y 7.

Nos damos cuenta de que el Sr. Picasso no sólo nació sino que murió. Incorporamos esa posibilidad, incluyendo además un separador opcional entre nacimiento y muerte. Dejamos de momento el tratamiento del "*n.*".

```
que --> leftB, lugar, comaOpcional, fecha, separador,  lugar,
        comaOpcional, fecha, rightB.

separador --> ['-'].
```

Además el predicado puede aparecer como una aposición (sin verbo copulativo) y el sintagma nominal no tiene por qué incluir un determinante.

```
predicado --> sn.
sn --> detOpcional, ns.
sn --> detOpcional, ns, adj.
```

```
detOpcional --> [].
detOpcional --> det.
```

falta alguna entrada del lexicón, como:

```
esCiudad('París').
esProfesion('teórico').
vc-->[fue].
```

Acabamos de colocar noRelevante en las posiciones que convengan:

```
oracion --> quien, noRelevante, que, comaOpcional, noRelevante,  predicado,
        noRelevante, punto.
que --> leftB, noRelevante, lugar, comaOpcional, fecha, rightB.
que --> leftB, noRelevante, lugar, comaOpcional, fecha, separador,
        lugar, comaOpcional, fecha, rightB.
```

Y las 7 oraciones quedan analizadas. Además hemos generalizado bastante la inclusión de los nuevos constituyente de forma que la gramática es bastante general.

Podemos mejorar la cosa haciendo que lugar y fecha puedan ser opcionales y aparecer en cualquier orden. Esto no es necesario pero si conveniente:

```
que --> leftB, noRelevante, lugarYOFecha, rightB.
que --> leftB, noRelevante, lugarYOFecha, separador,  lugarYOFecha, rightB.

lugarYOFecha --> lugar, comaOpcional, fecha.
lugarYOFecha --> lugar.
lugarYOFecha --> fecha.
lugarYOFecha --> fecha, comaOpcional, lugar.
```

c) El enunciado ha quedado un tanto machista (todos los artistas de los ejemplos son hombres). Qué cambios habría que realizar en gramática y lexicón para incorporar concordancia de género.

Realmente muy pocos, consideremos otro ejemplo:

"Magdalena Carmen Frida Kahlo y Calderón, mejor conocida como Frida Kalho (n. Coyoacán, Ciudad de México, 6 de julio de 1907 - 13 de julio de 1954) fue una destacada pintora mexicana."

Hay que añadir género a los nombres, adjetivos y determinantes y establecer la concordancia.

```
esProfesion('pintor',m).
esProfesion('pintora',f).
...
esNacionalidad('mexicana',f).
...
adj(m) --> [destacado].
adj(f) --> [destacada].
...
det(m) --> [un].
det(f) --> [una].
...
```

Sólo hay que controlar la concordancia en:

```
sn --> detOpcional(G), ns(G).
sn --> detOpcional(G), ns(G), adj(G).

ns(G) --> adjOpcional(G), n(G), adjOpcional(G).
ns(G) --> n1(G),conj,n(G).
```

```
n1(G) --> adjOpcional(G), n(G), adjOpcional(G).
n1(G) --> n(G),coma,n1(G).
```

Ademas deberíamos añadir el género en los nombres y la concordancia entre `quien` y `predicado`.

d) Deseamos obtener una representación del significado de las oraciones correctas del estilo siguiente: persona(nombre, fechaNacimiento, lugarNacimiento, fechaDefuncion, lugarDefuncion, actividad, nacionalidad)

donde las fechas se expresarían con el predicado fecha(dia, mes, any) y los lugares como lugar(ciudad, provincia, pais). La actividad se representaría como una lista. Los datos no existentes se representarían con nil.

Así para las 7 oraciones el resultado a extraer sería:

```
persona('Pablo_Ruiz_Picasso', fecha(25,'octubre',1881),
lugar('Málaga',nil,'España'),fecha(8,'abril',1973),
lugar('Mougins',nil,'Francia'),['pintor','escultor'] ,'español')

persona('Paul_Auster', fecha(3,'febrero',1947),
lugar('Newark','Nueva_Jersey',nil), nil,nil,
['escritor'], 'estadounidense')

persona('Pierre_Corneille', fecha(6,junio,1606),
lugar('Ruán', nil, 'Francia'),fecha(1,octubre,1684),
lugar('París',nil,nil),['dramaturgo'],'francés'

persona('Juan_Marsé_Carbó', fecha(8,'enero',1933),
lugar('Barcelona',nil,nil), nil, nil,['novelista'],'español')

persona('Jaume_Cabré_Fabré', fecha(nil, nil, 1947),
lugar('Barcelona',nil,nil), nil, nil,['filólogo','escritor'] ,'catalán' )

persona('Antoni_Tàpies_i_Puig', fecha(13,'diciembre',1923),
lugar('Barcelona',nil,nil), nil, nil,['pintor,', 'escultor'] ,'español')

persona('Antonio_López_García', fecha( 6,'enero',1936),
lugar('Tomelloso','Ciudad_Real',nil),
nil, nil, ['pintor', 'escultor'] ,'español')
```

¿Cómo deberíamos modificar la DCG para obtener el efecto deseado?

Es sumamente sencillo, si procedemos top down (también lo podeis hacer al revés, según lo explicado en clase):

```
oracion(persona(N,FN,LN,FD,LD,Ac,Nac)) --> quien(N), noRelevante, que(FN,LN,FD,LD),
comaOpcional, noRelevante,  predicado(Ac,Nac), noRelevante, punto.
```

Es decir hemos de proporcionar N[ombre], F[echa]N[acimiento], L[ugar]N[acimiento], F[echa]D[efuncion], L[ugar]D[efuncion], Ac[tividades],Nac[ionalidad]. Es obvio comprobar qué componente me debe proporcionar cada información.

Procedemos de igual forma top down:

```
quien(Q) --> [Q], {esPersona(Q)}, noRelevante .

que(FN,LN,FD,LD) --> leftB, noRelevante, lugarYOFecha(FN,LN), separador,
lugarYOFecha(FD,LD), rightB.
que(FN,LN,nil,nil) --> leftB, noRelevante, lugarYOFecha(FN,LN), rightB.
```

```
lugarYOFecha(L,F) --> lugar(L), comaOpcional, fecha(F).
lugarYOFecha(L,nil) --> lugar(L).
lugarYOFecha(nil,F) --> fecha(F).
lugarYOFecha(L,F) --> fecha(F), comaOpcional, lugar(L).


predicado(Ac,Nac) --> vc, sn(Ac,Nac).
predicado(Ac,Nac) --> sn(Ac,Nac).
```

Fijaros en que para construir la interpretación de lugar usamos el functor lugar con tres argumentos, es también sencillo ver qué constituyente proporciona cada argumento.

```
lugar(lugar(X,nil,nil)) --> [X],{esCiudad(X)}.
lugar(lugar(X,Y,nil)) --> [X],{esCiudad(X)}, comaOpcional, [Y],{esProvincia(Y)}.
lugar(lugar(X,nil,Y)) --> [X],{esCiudad(X)}, comaOpcional, [Y],{esPais(Y)}.

fecha(fecha(nil,nil,X)) --> [X],{esAny(X)}.
fecha(fecha(X,Y,Z)) --> [X],{esDia(X)}, separador, [Y],{esMes(Y)}, separador,
[Z],{esAny(Z)}.

sn(Ac,Nac) --> detOpcional(G), ns(G,Ac), adj(G,Nac).
sn(Ac,nil) --> detOpcional(G), ns(G,Ac).

ns(G,[A]) --> adjOpcional(G), n(G,A), adjOpcional(G).
ns(G,[A|LA]) --> n1(G,LA),conj,n(G,A).
n1(G,[A]) --> adjOpcional(G), n(G,A), adjOpcional(G).
n1(G,[A|LA]) --> n(G,A),coma,n1(G,LA).
```

Algún cambio (pocos) es necesario en el lexicón:

```
n(G,N) --> [N], {esProfesion(N,G)}.
adj(G,A) --> [A], {esNacionalidad(A,G)}.
adj(m,nil) --> [destacado].
adj(f,nil) --> [destacada].
```

Aquí teneis el resultado del proceso:

```
oracion 1 analizada correctamente:
persona(Pablo_Ruiz_Picasso,fecha(25,octubre,1881),lugar(Málaga,nil,España),
fecha(8,abril,1973),lugar(Mougins,nil,Francia),[escultor,pintor],español)

oracion 2 analizada correctamente:
fecha(3,febrero,1947),persona(Paul_Auster,lugar(Newark,Nueva_Jersey,nil),
nil,nil,[escritor],estadounidense)

oracion 3 analizada correctamente:
persona(Pierre_Corneille,fecha(6,junio,1606),lugar(Ruán,nil,Francia),
fecha(1,octubre,1684),lugar(París,nil,nil),[dramaturgo],francés)

oracion 4 analizada correctamente:
persona(Juan_Marsé_Carbó,fecha(8,enero,1933),lugar(Barcelona,nil,nil),nil,nil,
[novelista],español)

oracion 5 analizada correctamente:
persona(Jaume_Cabré_Fabré,fecha(nil,nil,1947),lugar(Barcelona,nil,nil),nil,nil,[escritor,
filólogo ] ,
catalán)
```

```
oracion 6 analizada correctamente:
persona(Antoni_Tàpies_i_Puig,fecha(13,diciembre,1923),lugar(Barcelona,nil,nil),
nil,nil,[pintor],nil)

oracion 7 analizada correctamente:
persona(Antonio_López_García,fecha(6,enero,1936),lugar(Tomelloso,Ciudad_Real,nil)
,nil,nil,[escultor,pintor],español)

oracion 8 analizada correctamente:
persona(Magdalena_Carmen_Frida_Kahlo_y_Calderón,fecha(6,julio,1907),
lugar(Coyoacán,nil,México),fecha(13,julio,1954),nil,[pintora],
mexicana)
```

3. Considereu la gramàtica i lexicó següents:

   GRAMÀTICA

   ```
   frase → sn, verbi.
   frase → sn, verbt(X), compl(X).
   sn → det, npr.
   sn → det, nom.
   compl([]) → [].
   compl([arg(nul)]) → sn.
   compl([arg(X) | Y]) → prep(X), sn, compl(Y).
   verbi → [W], {vi(W)}.
   verbt(A) → [W], {vt(W,A)}.
   npr → [W], {np(W)}.
   nom → [W], {n(W)}.
   det → [W], {d(W)}.
   prep(W) → [W], {p(W)}.
   ```

   LEXICÓ

   ```
   np(clara).
   np(maria).
   np(joan).
   n(llibre).
   n(mestra).
   n(gata).
   d(un).
   d(una).
   d(el).
   d(la).
   d(en).
   p(en).
   p(a).
   p(amb).
   p(de).
   vi(corre).
   vi(riu).
   vt(llegeix,[arg(nul)]).
   vt(parla,[arg(de),arg(amb)]).
   vt(pensa,[arg(en)]).
   ```

   Responeu en ordre les següents questions. Atenció! Totes les extensions han de ser incrementals. NO cal que escriviu cada vegada TOTA la gramàtica i el lexicó. Només cal que indiqueu les coses que

canviaríen a cada pas. Si cal, podeu suposar que teniu definits predicats PROLOG per al tractament de llistes.

a) Digueu quines de les següents frases la gramàtica reconeix com a correctes i quines considera errònies.
1. La Clara llegeix un llibre
2. El Clara llegeix una llibre
3. La Clara llegeix
4. La Maria parla amb en Joan de la gata
5. La Maria parla amb en Joan
6. La Maria parla
7. El llibre parla de la gata amb en Joan
8. El llibre riu

Las frases 3, 4, 5 y 6 son rechazadas por la gramática. El problema esta en que o no aparecen los complementos necesarios o no están en el orden establecido.

b) Exteneu la gramàtica i/o el lexicó per a que pugui tractar llistes de "sn", com per exemple:
En Joan, la Maria i la Clara llegeixen un llibre
La mestra parla de la Maria i en Joan amb la Clara
NO us preocupeu aquí de les contraccions: "de + en" = "d'en", etc. Atenció! cal tenir en compte la concordança de nombre, però NO us preocupeu del gènere.

Primero debemos añadir entradas para los verbos en el lexicón que sean singular y plural y añadir esa información en cada palabra.

```
vi(corre,s).
vi(riu,s).
vi(corren,p).
vi(riuen,p).
vt(llegeix,s,[arg(nul)]).
vt(parla,s,[arg(de),arg(amb)]).
vt(pensa,s,[arg(en)]).
vt(llegeixen,p,[arg(nul)]).
vt(parlen,p,[arg(de),arg(amb)]).
vt(pensen,p,[arg(en)]).
```

También hemos de modificar los terminales de los verbos para que pasen esta información a las reglas:

```
verbi(C) --> [W], {vi(W,C)}.
verbt(C,A) --> [W], {vt(W,C,A)}.
```

Ahora debemos modificar las reglas del sintagma nominal de manera que admita además una lista de sintagmas nominales e incluya la información del número:

```
sn(s) --> ilsn.
sn(p) --> lsn.

lsn --> ilsn, conj, ilsn.
lsn --> ilsn, coma, lsn.

ilsn --> det, npr.
ilsn --> det, nom.
```

Añadiremos las nuevas categorias sintácticas a los terminales y al lexicón:

```
conj --> [W],{cj(W)}.
coma --> [W],{cm(W)}.
cj(i).
cm(',').
```

Finalmente hemos de modificar las reglas que deben comprobar la concordancia en número:

```
frase --> sn(C), verbi(C).
frase --> sn(C), verbt(C,X), compl(X).
```

Dado que en los complementos no se tiene que comprobar la concordancia podemos añadir una variable anónima que oculte el parámetro adicional.

```
compl([arg(nul)]) --> sn(_).
compl([arg(X)|Y]) --> prep(X), sn(_), compl(Y).
```

c) Exteneu la gramàtica i/o el lexicó per a que comprovi les següents restriccions semàntiques dels verbs:
correr: subjecte ANIMAT
llegir: subjecte HUMÀ i complement de tipus INANIMAT
pensar: subjecte HUMÀ
parlar: subjecte i complement introduit per "amb" HUMANS
riure : subjecte HUMÀ
Doneu-vos compte de que una entitat HUMANA es també ANIMADA.
Així, "En Joan i la Clara riuen" i "La gata corre" són correctes, però "En Joan i la gata riuen" i "El llibre corre" han de ser considerades incorrectes.

Para comprobar las restricciones semánticas hemos de añadir esta información a los nombres y a los verbos en el lexicón:

```
np(clara,[huma,animat]).
np(maria,[huma,animat]).
np(joan,[huma,animat]).
n(llibre,[inanimat]).
n(mestra,[huma,animat]).
n(gata,[animat]).

vi(corre,s,asem(animat)).
vi(riu,s,asem(huma)).
vi(corren,p,asem(animat)).
vi(riuen,p,asem(huma)).
vt(llegeix,s,[arg(nul)],asem(huma,[inanimat])).
vt(parla,s,[arg(de),arg(amb)],asem(huma,[_,huma])).
vt(pensa,s,[arg(en)],asem(huma,[_])).
vt(llegeixen,p,[arg(nul)],asem(huma,[inanimat])).
vt(parlen,p,[arg(de),arg(amb)],asem(huma,[_,huma])).
vt(pensen,p,[arg(en)],asem(huma,[_])).
```

Hay que notar que algunos nombres pueden tener más de una categoría semántica. Para los verbos, con la función `asem` incluimos la restricción semántica del sujeto como primer parámetro y en el caso de los verbos transitivos añadimos como segundo parámetro una lista de restricciones semánticas para cada complemento. En el caso de no haber restricción semántica para algún elemento, usamos una variable anónima.

También deberemos modificar los terminales para extraer esta información del lexicón y pasársela a las reglas de la gramática:

```
verbi(C,S) --> [W], {vi(W,C,S)}.
verbt(C,A,S) --> [W], {vt(W,C,A,S)}.
npr(S) --> [W], {np(W,S)}.
nom(S) --> [W], {n(W,S)}.
```

Las reglas del sintagma nominal también deberán modicarse para pasar la información de las categorías semánticas de sus elementos:

```
sn(s,[S]) --> ilsn(S).
sn(p,S) --> lsn(S).

lsn([X|[Y]]) --> ilsn(X), conj, ilsn(Y).
lsn([X|[Y]]) --> ilsn(X), coma, lsn(Y).

ilsn(S) --> det, npr(S).
ilsn(S) --> det, nom(S).
```

Para la comprobación de la restricción semántica comprobaremos que la categoría indicada para el verbo esté dentro de la lista de categorías del sintagma nominal. Dado que un sintagma nominal puede tener varios nombres deberemos comprobar que para cada elemento se cumple la restricción. Para probar eso introducimos un predicado prolog:

```
check(_,[]).
check(X,[Y|Z]):- member(X,Y), check(X,Z).
```

Este comprueba para cada elemento de la lista del sintagma nominal que la categoría semántica este entre las de la del elemento.

La restricción semántica la deberemos comprobar en la producción frase para el sujeto:

```
frase --> sn(C,S1), verbi(C,asem(S2)),{check(S2,S1)}.
frase --> sn(C,S1), verbt(C,X,asem(S2,S3)),{check(S2,S1)}, compl(X,S3).
```

Y en la producción del complemento:

```
compl([],[])--> [].
compl([arg(nul)],[S1]) --> sn(_,S2),{check(S1,S2)}.
compl([arg(X)|Y],[S1|S]) --> prep(X), sn(_,S2),{check(S1,S2)}, compl(Y,S).
```

6. Considerad las siguientes oraciones:

1	hoteles en Barcelona
2	hoteles cerca de Barcelona
3	casas de colonias al norte de Barcelona
4	hoteles entre Barcelona y Tarragona
5	hoteles a 100 km de Barcelona
6	casas en el centro de Barcelona
7*	casas en la centro de Barcelona
8*	hoteles a 100 kg de Barcelona
9*	hoteles entre Barcelona y Barcelona
10*	hoteles entre Barcelona
11*	Barcelona entre hoteles

Las 6 primeras son correctas y debieran ser analizadas por un analizador, las 5 últimas son erróneas y debieran ser rechazadas.

Considerad ahora la siguiente DCG:

```
oracion --> que, rel_geo, donde.
que --> sn.
donde --> prep, sn.
sn_bas --> n.
sn_bas -->det, n.
sn_bas --> npr.
sn --> sn_bas, sp.
sn --> sn_bas.
sp --> prep, sn.
```

```
n --> [X],{es_n(X)}.
npr --> [X],{es_npr(X)}.
n --> [X],{es_npr(X)}.
prep --> [X],{es_prep(X)}.
det --> [X], {es_det(X)}.
```

Con el siguiente lexicon:

```
es_npr('Barcelona').
es_npr('Tarragona').
es_n('hotel').
es_n('hoteles').
es_n('casa').
es_n('casas').
es_n('colonias').
es_n('centro').
es_prep('a').
es_prep('de').
es_prep('entre').
es_prep('en').
es_det('un').
es_det('el').
es_unidad('km').
es_unidad('kg').
es_numero(X):-  number(X).

rel_geo --> [].
rel_geo --> [cerca].
rel_geo --> prep,[X],{es_punto_cardinal(X)}.

es_punto_cardinal(norte).
```

Se pide :

a) Decid qué oraciones analizaría la DCG, correctamente si son de las 6 primeras, e incorrectamente en caso contrario.

Considerad las oraciones del ejemplo descritas como sigue:

```
or(1,['hoteles','en','Barcelona'],ok).
or(2,['hoteles','cerca','de','Barcelona'],ok).
or(3,['casas','de','colonias','al','norte','de','Barcelona'],ok).
or(4,['hoteles','entre','Barcelona','y','Tarragona'],ok).
or(5,['hoteles','a','100','km','de','Barcelona'],ok).
or(6,['casas','en','el','centro','de','Barcelona'],ok).
or(7,['casas','en','la','centro','de','Barcelona'],ko).
or(8,['hoteles','a','100','kg','de','Barcelona'],ko).
or(9,['hoteles','entre','Barcelona','y','Barcelona'],ko).
or(10,['hoteles','entre','Barcelona'],ko).
or(11,['Barcelona','entre','hoteles'],ko).
```

considerad también las siguientes cláusulas para analizarlas:

```
analizar(X):-
  or(X,Or,ok),
  oracion(Or,[]),
  print('oracion '),print(X),print(' analizada correctamente'),nl.
```

```
analizar(X):-
  or(X,Or,ko),
  oracion(Or,[]),
  print('oracion '),print(X),print(' analizada incorrectamente'),nl.

analizar(X):-
  or(X,_,ko),
  print('oracion '),print(X),print(' correctamente no analizada'),nl.

analizar(X):-
  or(X,_,ok),
  print('oracion '),print(X),print(' incorrectamente no analizada'),nl.

analizar_all:-
  findall(X,or(X,_,_),XX),
  analizarl(XX).

analizarl([]).
analizarl([X|Y]):-
  analizar(X),
  analizarl(Y).
```

El resultado es el siguiente:

```
oracion 1 analizada correctamente
oracion 2 analizada correctamente
oracion 3 incorrectamente no analizada
oracion 4 incorrectamente no analizada
oracion 5 incorrectamente no analizada
oracion 6 analizada correctamente
oracion 7 correctamente no analizada
oracion 8 correctamente no analizada
oracion 9 correctamente no analizada
oracion 10 analizada incorrectamente
oracion 11 analizada incorrectamente
```

b) Qué modificaciones o adiciones deberían incorporarse a la gramática y lexicón para que la DCG funcionara correctamente sobre las 11 oraciones (y otras similares).

analizar la 3 es simplemente añadir la preposición 'al' al lexicón.

```
es_prep('al').
```

para analizar la 4 hemos de considerar relaciones geográficas binarias (entre .. y):

```
oracion --> que, rel_geo_2_1, sn, rel_geo_2_2, sn.
```

dónde

```
rel_geo_2_1 --> [X],{rel_geo_2(X,_)}.
rel_geo_2_2 --> [X],{rel_geo_2(_,X)}.
```

y

```
rel_geo_2(entre,y).
```

para analizar la 5 debemos incluir

```
rel_geo --> prep,[X,Y],{es_numero(X),es_unidad(Y)}.
```

Para ser honestos hay que reconocer que para que la oración 5 se analice en vez de '100' deberíais poner 100, en caso contrario el predicado number no se satisface, una alternativa es añadir:

```
es_numero(X):-
  name(X,Y),
  number_chars(Z,Y),
  number(Z).
```

Con todos estos cambios las oraciones 1 a 5 se analizan correctamente, pero también se analizan algunas que deberían rechazarse:

```
oracion 1 analizada correctamente
oracion 2 analizada correctamente
oracion 3 analizada correctamente
oracion 4 analizada correctamente
oracion 5 analizada correctamente
oracion 6 analizada correctamente
oracion 7 correctamente no analizada
oracion 8 analizada incorrectamente
oracion 9 analizada incorrectamente
oracion 10 analizada incorrectamente
oracion 11 analizada incorrectamente
```

Para más inri podemos darnos cuenta de que la oración 7 no se analiza simplemente porque en el lexicón no existe la entrada 'al',
si la anadimos:

```
es_det('la').
```

las oraciones de la 7 a la 11 son ahora analizadas y deberían ser rechazadas.
Simplemente eliminando del lexicón la entrada

```
es_prep('entre').
```

es decir exigiendo que la aparición de 'entre' vaya condicionada a la de 'y' podemos solucionar las oraciones 10 y 11.
Para solucionar la 8 deberíamos exigir que la unidad de medida sea de longitud y no de peso:

```
es_unidad('km',longitud).
es_unidad('kg',masa).

rel_geo --> prep,[X,Y],{es_numero(X),es_unidad(Y,longitud)}.
```

Para solucionar la 9 debemos exigir que los dos lugares sean diferentes, para ello el sn debe tener un parámetro que debe contener el nombre. Ello obliga a cambiar bastantes cláusulas:

```
oracion --> que, rel_geo_2_1, sn(X), rel_geo_2_2, sn(Y),{X\==Y}.
que --> sn(_).
donde --> prep, sn(_).
sn_bas(X) --> n(X).
sn_bas(X) -->det, n(X).
sn_bas(X) --> npr(X).
sn(X) --> sn_bas(X), sp.
sn(X) --> sn_bas(X).
sp --> prep, sn(_).


n(X) --> [X],{es_n(X)}.
npr(X) --> [X],{es_npr(X)}.
n(X) --> [X],{es_npr(X)}.
```

aunque, como veis, los cambios en cada cláusula son mínimos
Finalmente para corregir la 7 basta establecer la concordancia en género (de paso comprobaremos la de número aunque no se nos pide)

```
sn_bas(X) -->det(G,N), n(X,G,N).
n(X,G,N) --> [X],{es_n(X,G,N)}.
det(G,N) --> [X], {es_det(X,G,N)}.

es_n('hotel',m,s).
es_n('hoteles',m,p).
es_n('casa',f,s).
es_n('casas',f,p).
es_n('colonias',f,p).
es_n('centro',m,s).
```

Ahora sí el resultado es el esperado:

```
oracion 1 analizada correctamente
oracion 2 analizada correctamente
oracion 3 analizada correctamente
oracion 4 analizada correctamente
oracion 5 analizada correctamente
oracion 6 analizada correctamente
oracion 7 correctamente no analizada
oracion 8 correctamente no analizada
oracion 9 correctamente no analizada
oracion 10 correctamente no analizada
oracion 11 correctamente no analizada
```

c) Deseamos obtener una representación del significado de las oraciones correctas del estilo siguiente: `busco(<qué>,<relación geográfica>,[<lista de localidades referidas>]).`

1	hoteles en Barcelona	`busco(hoteles,en,['Barcelona'])`
2	hoteles cerca de Barcelona	`busco(hoteles,cerca,['Barcelona'])`
3	casas de colonias al norte de Barcelona	`busco([casas,de,colonias],norte,['Barcelona'])`
4	hoteles entre Barcelona y Tarragona	`busco(hoteles,entre,['Barcelona'Tarragona'])`
5	hoteles a 100 km de Barcelona	`busco(hoteles,[100,km],['Barcelona'])`
6	casas en el centro de Barcelona	`busco(casas,centro,['Barcelona'])`

¿Cómo deberíamos modificar la DCG para obtener el efecto deseado?

Es bastante sencillo:

```
oracion(busco(Q,R,D)) --> que(Q), donde(R,D).
oracion(busco(Q,R,D)) --> que(Q), rel_geo(R), donde(_,D).
oracion(busco(Q,[R],[X,Y])) --> que(Q), rel_geo_2_1(R), sn(X), rel_geo_2_2,
                                sn(Y),{X\==Y}.
```

recupero la fórmula deseada, busco(Q,R,D), desgloso la primera cláusula en dos para permitir recoger la preposición como relación geográfica, elimino

```
/*rel_geo([]) --> [].*/
```

```
donde(P,X) --> prep(P), sn(X).
```

devuelve la relación geográfica y el lugar

```
rel_geo_2_1(X) --> [X],{rel_geo_2(X,_)}.
```

caso de 'entre' ...

```
prep([X]) --> [X],{es_prep(X)}.
```

El resultado es:

```
oracion 1 analizada correctamente: busco(hoteles,[en],Barcelona)
oracion 2 analizada correctamente: busco(hoteles,[cerca],Barcelona)
oracion 3 analizada correctamente: busco(casas,[norte],Barcelona)
oracion 4 analizada correctamente: busco(hoteles,[entre],[Barcelona,Tarragona])
oracion 5 analizada correctamente: busco(hoteles,[100,km],Barcelona)
oracion 6 analizada correctamente: busco(casas,[de],Barcelona)
oracion 7 correctamente no analizada
oracion 8 correctamente no analizada
oracion 9 correctamente no analizada
oracion 10 correctamente no analizada
oracion 11 correctamente no analizada
```

19. Estas navidades Santa Claus ha decidido hacer más fácil el hacer llegar las peticiones de regalos y quiere crear un servicio telefónico donde uno puede dejar un mensaje con sus deseos. Los elfos de Santa no saben inteligencia artificial, así que eres tú el que tiene que definir el sistema de procesamiento del lenguaje natural para procesar los mensajes.

    El tipo de peticiones que pueden llegarnos son de este estilo:

    - 3 cajas de bombones
    - 3 bolsas de caramelos Suchard
    - 6 cajas de galletas
    - 5 frascos de colonia Eau de Rochas
    - 5 kilos de carbón
    - 1 kilo de polvorones
    - 6 corbatas
    - 3 muñecas Barbie
    - 1 bibicleta

    a) Define una DCG capaz de reconocer frases de este tipo. Asume que tienes un predicado Prolog `number(X)` que es cierto si `X` es un número.

En las oraciones que queremos reconocer tenemos dos patrones, uno en el que aparece la cantidad, las unidades y el tipo de regalo y otro en el que no aparecen las unidades:

```
peticion --> cantidad, unidades, regalo.
peticion --> cantidad, regalo.
```

Analizar la cantidad es muy sencillo:

```
cantidad --> [W],{number(W)}
```

Las unidades también son muy sencillas

```
unidades --> [W],{unidad(W)}
```

La estructura del regalo es variable, puede aparecer la preposicion *de* o no y el tipo de regalo puede incluir la marca. Podemos analizarlo con las siguientes reglas:

```
regalo --> prep, producto, marca.
regalo --> producto, marca.
```

ahora necesitamos los siguientes terminales:

```
prep -> [de].
producto -> [W],{prod(W)}.
marca -> [].
marca -> [W],{marcas([W])}.
marca -> [W1,W2,W3],{marcas([W1,W2,W3])}.
```

Con esto tenemos la siguiente gramática:

```
peticion --> cantidad, unidades, regalo.
peticion --> cantidad, regalo.
regalo --> prep, producto, marca.
regalo --> producto, marca.
prep -> [de].

cantidad --> [W],{number(W)}
unidades --> [W],{unidad(W)}
producto -> [W],{prod(W)}.
marca -> [].
marca -> [W],{marcas([W])}.
marca -> [W1,W2,W3],{marcas([W1,W2,W3])}.
```

A la que le podemos añadir el siguiente lexicón:

```
unidad(caja).
unidad(cajas).
unidad(bolsa).
unidad(bolsas).
unidad(kilo).
unidad(kilos).
...

prod(bombon)
prod(bombones)
prod(caramelo)
prod(caramelos)
prod(carbón)
prod(polvorones)
prod(corbata)
prod(corbatas)
prod(muñeca)
prod(muñecas)
...

marcas([suchard]).
marcas([barbie]).
marcas([eau,de,rochas]).
```

b) Modifica la gramática para que tenga en cuenta que la cantidad ha de coordinar en número con las unidades del producto o el producto.

Primero hemos de añadir la información del número en las entradas del lexicón donde corresponda:

```
unidad(caja,singular).
unidad(cajas,plural).
unidad(bolsa,singular).
unidad(bolsas,plural).
unidad(kilo,singular).
unidad(kilos,plural).
...

prod(bombon,singular)
prod(bombones,plural)
```

```
prod(caramelo,singular)
prod(caramelos,plural)
prod(corbata,singular)
prod(corbatas,plural)
...
```

También hemos de modificar la reglas que corresponden a los terminales:

```
cantidad(singular) --> [W],{number(W),W=1}
cantidad(plural) --> [W],{number(W),>=1}
unidades(N) --> [W],{unidad(W,N)}
producto(N) -> [W],{prod(W,N)}.
```

Y ahora hemos de propagar la información en las reglas de la gramática:

```
peticion --> cantidad(N), unidades(N), regalo(_).
peticion --> cantidad(N), regalo(N).
regalo(N) --> prep, producto(N), marca.
regalo(N) --> producto(N), marca.
```

c) Modifica la gramática para que las unidades se correspondan con el tipo de regalo que se pide.

Primero hemos de añadir la información sobre la coherencia de unidades en el lexicón. En este caso hay productos que van en unas unidades específicas y otros que no tienen unidades. En este ultimo caso utilizamos la constante `ninguna` como unidades.

En el caso de que consideremos que algunos regalos se pueden pedir con o sin unidades podemos añadir las entradas del lexicón que lo permitan. Si algunos regalos los admitimos con mas de unas unidades podemos simplemente replicar las entradas.

Si queremos una solución más limpia podemos usar listas y comprobar que las unidades pertenezcan a esa lista.

```
unidad(caja,singular,caja).
unidad(cajas,plural,caja).
unidad(bolsa,singular,bolsa).
unidad(bolsas,plural,bolsa).
unidad(kilo,singular,kilo).
unidad(kilos,plural,kilo).
...

prod(bombon,singular,caja)
prod(bombones,plural,caja)
prod(caramelo,singular,bolsa)
prod(caramelos,plural,bolsa)
prod(corbata,singular,ninguna)
prod(corbatas,plural,ninguna)
...
```

Ahora cambiamos los terminales correspondientes:

```
unidades(N,U) --> [W],{unidad(W,N,U)}
producto(N,U) -> [W],{prod(W,N,U)}.
```

Y finalmente las reglas de la gramática:

```
peticion --> cantidad(N), unidades(N,U), regalo(_,U).
peticion --> cantidad(N), regalo(N,ninguna).
regalo(N,U) --> prep, producto(N,U), marca.
regalo(N,U) --> producto(N,U), marca.
```

d) Modifica la gramática para que genere como salida para cada petición el predicado `Regalo(Cantidad,Unidades,Regalo)`

El generar la salida necesita que demos una representación a cada entidad con significado, para la cantidad usaremos el número, para las unidades podemos usar la información que hemos añadido para el apartado anterior, para el regalo deberemos unir la información del tipo de regalo y la marca.

Para la representación de estos dos últimos elementos podemos añadir una nueva entrada a estas categorías o usar las palabras que encontramos. Para los regalos que no especifican marca podemos usar la constante `cualquiera`

Primero tocamos el lexicón:

```
prod(bombon,singular,caja,bombon)
prod(bombones,plural,caja,bombon)
prod(caramelo,singular,bolsa,caramelo)
prod(caramelos,plural,bolsa,caramelo)
prod(corbata,singular,ninguna,corbata)
prod(corbatas,plural,ninguna,corbata)
...

marcas([suchard],suchard).
marcas([barbie],barbie).
marcas([eau,de,rochas],eauderochas).
```

ahora tocamos los terminales:

```
cantidad(singular,W) --> [W],{number(W),W=1}
cantidad(plural,W) --> [W],{number(W),>=1}
producto(N,U,S) -> [W],{prod(W,N,U,S)}.
marca(cualquiera) -> [].
marca(S) -> [W],{marcas([W],S)}.
marca(S) -> [W1,W2,W3],{marcas([W1,W2,W3],S)}.
```

Y finalmente tocamos las reglas de la gramática:

```
peticion(regalo(C,U,R)) --> cantidad(N,C), unidades(N,U), regalo(_,U,R).
peticion(regalo(C,ninguna,R)) --> cantidad(N,C), regalo(N,ninguna,R).
regalo(N,U,reg(P,M)) --> prep, producto(N,U,P), marca(M).
regalo(N,U,reg(P,M)) --> producto(N,U,P), marca(M).
```

7.1 Búsqueda

7.1.1 Como plantear los problemas cuestiones de búsqueda

En estos problemas se plantean diferentes alternativas que pretenden solucionar el problema planteado. Estas alternativas utilizan los tres mecanismos de resolución de problemas que se han visto en teoría: algoritmos de búsqueda heurística, algoritmos de búsqueda local y algoritmos de satisfacción de restricciones.

Para criticar cada alternativa se han de tener claras las características de cada uno de estos tres mecanismos:

Búsqueda heurística El problema se ha de plantear como la construcción de un camino, se definirá un estado inicial y un estado final que cumpla las características del problema.

Serán los operadores los que partiendo del estado inicial vayan construyendo el camino completo que llegue al estado final. Estos operadores han de comprobar las restricciones del problema y han de dar estados válidos. El factor de ramificación ha de ser el adecuado y no ha de haber mas o menos operadores de los necesarios.

La función heurística ha de dirigir la búsqueda hacia el tipo de solución que pide el problema. También ha de ser admisible y suficientemente informada como para garantizar una solución en un tiempo razonable. Si no hay posibilidad de encontrar un heurístico adecuado este tipo de resolución de problemas no sería viable debido a su coste a pesar de que permita encontrar una solución.

Búsqueda local El problema se ha de plantear como la mejora de una solución inicial. La solución incial por lo general deberá cumplir las restricciones a no ser que sea demasiado costoso. Si no se cumplen las condiciones de solución, se ha de garantizar que la búsqueda acabará en el espacio de soluciones.

Los operadores han de comprobar las restricciones del problema y generar soluciones correctas. Los operadores han de ser los necesarios y su factor de ramificación debe ser el justo para permitir la exploración del espacio de búsqueda.

La función heurística ha de incluir todos los elementos que se quieren optimizar y todos han de ir en la dirección adecuada. Si existen ponderaciones entre los diferentes elementos estas deberían estar justificadas.

Satisfacción de restricciones El problema se ha de plantear como la asignación de un conjunto de valores a unas variables. La elección de que son variables y sus dominios ha de ser correcta.

El conjunto de restricciones ha de ser completo y correcto.

El problema no ha de buscar la optimización de ningún parámetro ya que estos algoritmos no lo permiten.

7.1.2 Problemas solucionados

7. Queremos determinar cómo abastecer N tiendas a partir de k almacenes. Tiendas y almacenes se encuentran distribuidos dentro de una ciudad y conocemos las coordenadas en las que se ubican. El abastecimiento se realiza mediante camiones que cargan las mercancías necesarias para una tienda, las llevan y vuelven al almacén. Cada tienda tiene una demanda ($D(T_i)$) diferente.

   Lo que nos interesa es que los almacenes tengan aproximadamente la misma cantidad de mercancías, por lo que la suma de las demandas de las tiendas que servimos desde cada almacén ha de ser apro-

ximadamente igual. También nos interesa que los kilómetros que han de recorrer los camiones para abastecer a las N tiendas sean los menos posibles.

Comenta cada una de las posibilidades indicando si resuelven o no el problema, qué errores te parece que tiene cada solución y cómo se podrían corregir, y qué ventajas e inconvenientes tienen cada una de ellas. Justifica la respuesta.

a) Queremos resolver el problema utilizando búsqueda local generando como solución inicial un estado en el que las tiendas están asignadas al almacén más cercano. Como operadores de búsqueda utilizamos cambiar una tienda de un almacén a otro. Como función heurística utilizamos la función:

$$h'(n) = \sum_{i=1}^{k} \sum_{\forall T_j \in Abast(A_i)} Dist(T_j, A_i) + \sum_{i=1}^{k} \left| \frac{\sum_{\forall T_j} D(T_j)}{k} - \sum_{\forall T_j \in Abast(A_i)} D(T_j) \right|$$

Donde $Abast(A_i)$ es el conjunto de tiendas que abastece el almacén A_i, $Dist(T_j, A_i)$ es la distancia entre la tienda T_j y el almacén A_i

L'estat inicial és perfectament correcte, tot i que encara podria estar més informat, ja que té en compte la optimització referent a les distàncies però no la donada per la distribució equitativa de les mercaderies entre magatzems.

Pel que fa a l'operador, ens permet recórrer tot l'espai de solucions: Podem visitar qualsevol estat en el que el nombre d'assignacions sigui el mateix que el de l'estat inicial, així doncs, donat que en l'estat inicial assignem cada botiga a algun magatzem, podem arribar a la solució. El factor de ramificació és $N \cdot (k-1)$, ja que per a cada botiga el podem assignar a qualsevol magatzem excepte l'actual. És un factor de ramificació alt però admissible.

Estudiem la funció heurística: El primer terme,

$$\sum_{i=1}^{k} \sum_{\forall T_j \in Abast(A_i)} Dist(T_j, A_i)$$

és el total de kilòmetres que s'hauran de recórrer els camions. Un terme que cal minimitzar. El segon terme,

$$\sum_{i=1}^{k} \left| \frac{\sum_{\forall T_j} D(T_j)}{k} - \sum_{\forall T_j \in Abast(A_i)} D(T_j) \right|$$

és la suma de les desviacions de les mercaderies d'un magatzem respecte la mitjana. Així doncs, també és un terme que cal minimitzar.

L'heurístic ho té tot en compte i sembla que la combinació dels elements és correcta, excepte per un detall: El primer dels elements és una suma de distàncies així que és una longitud. El segon és una suma de desviacions de mercaderies, així que s'interpreta com un valor en alguna unitat adient (de massa, volum...). En qualsevol cas, estem sumat dues quantitats que caldria ponderar per a què en el resultat els dos termes siguin significatius (Pensem què passaria si el total de quilòmetres fós de l'ordre de 100km i les mercaderies de 100.000kg; el primer valor no es tindria pràcticament en compte durant la búsqueda). Això, però, és ben senzill d'arreglar: n'hi ha prou afegint una constant en un dels dos sumands que faci que l'ordre de magnitud dels dos termes sigui similar, o un una mica major que l'altre, intencionadament, si li volem donar més importància. Una altra opció seria combinar-los mitjançant el producte, que fa que el resultat sigui més dependent dels dos termes, però perdem el control de la ponderació (que ara es podria fer amb exponents, però la interpretació del resultat és menys natural).

Així doncs, llevat de la possiblement necessària ponderació dels sumands de l'heurístic, el plantejament és correcte i l'aplicació d'un algorisme de cerca local ben parametritzat ens donarà la solució òptima o alguna pròxima a aquesta.

b) Queremos solucionar el problema mediante satisfacción de restricciones, como variables escogemos $k \times N$ variables binarias de manera que cada tienda tiene asociadas k variables que representan el

almacén al que está asignada. Como restricción imponemos que solo una de esas variables puede ser cierta. Añadimos también las siguientes restricciones al problema:

- Para cualquier almacén, la suma de las demandas de las tiendas asignadas a él no puede ser mayor en un 5 % a la suma de las demandas de las tiendas asignadas a cualquiera otro almacén.

$$\forall i, j (i \neq j) \sum_{\forall T_a \in Abast(A_i)} D(T_a) \leq \sum_{\forall T_b \in Abast(A_j)} D(T_b) \times 1.05$$

- Para cualquier almacén, la suma de las distancias de las tiendas asignadas a él no puede ser mayor en un 5 % a la suma de las distancias de las tiendas asignadas a cualquiera otro almacén.

$$\forall i, j (i \neq j) \sum_{\forall T_a \in Abast(A_i)} Dist(T_a, A_i) \leq \sum_{\forall T_b \in Abast(A_j)} Dist(T_b, A_j) \times 1.05$$

Com sabem, la definició d'un problema per a la seva resolució mitjançant satisfacció de restriccions consisteix en un conjunt de variables, amb els seus corresponents dominis, i un conjunt de restriccions entre els dominis d'aquestes variables, és a dir, un conjunt de restriccions cadascuna de les quals afecta a, com a màxim, dues variables. Abans de res, cal veure si l'esquema plantejat segueix aquest patró.

Ens defineixen un seguit de variables, X_{ij} $i = 1...k$, $j = 1...N$ amb dominis $X_{ij} \in \{0, 1\}$. La primera restricció diu que dues d'aquestes variables, referents al mateix magatzem no poden ser certes alhora, i.e., $\forall i = 1...k,\ X_{ij_0} = 1 \Rightarrow X_{ij} = 0\ \forall j \neq j_0$.

Les altres restriccions defineixen les restriccions que han de cumplir els magatzems respecte a la seva demanda i les seves distancias com un percentatje que preten mantenir cer equilibri entre aquestes quantitats.

Tal com estan definides les variables es relativament complicat fer el calcul d'aquestes restriccions. Un plantejament mes adient seria definir com variables els magatzems i com dominis les botigues. D'aquesta manera cada magatzem tindria un conjunt de botigues i tindriem que posar la restricción de que una botiga només podría apareixer a un magatzem.

Aquest plantejament permetria un cálcul mes senzill de les altres dues restriccions.

Si ens fixem en la restricció sobre la demanda de les botigues asignades a un magatzem veiem que la restricciá demana que la seva suma no sigui mes d'un 5 % superior que la de la resta. Això, encara que equilibra les demandes no es exactament el que demana el problema. De fet no sabem si es pot cumplir o si es pot arribar a una solución on la diferencia entre demandes sigui mes petita.

Si ens fixem en la restricción sobre les distàncies veiem que la restrición demana el mateix que per les demandes i el problema el que demana es que la distancia sigui mínima. Aquesta restricción equilibrara les distancies, pero no ens asegurara que la suma de distancies sigui mínima.

Una manera d'arreglar parcialment el problema seria posar una restricción que imposses que la suma de les distancies fos mes petita que certa quantitat limit. El problema d'això es que hauriem de coneixer a priori quin es el límit que es pot cumplir.

Altre problema amb posar una restricción amb un límit es que donat que la restricció es una suma de totes les distancies, es díficil de comprobar i ens obligaría a fer una búsqueda molt costosa perque es difícil aprofitar les avantatges del algoritmes que propagan restriccions com per exemple forward checking.

Resumint, donat el que ens demana es problema no es una bona idea plantejar-lo com un problema de satisfaccions de restriccions.

11. El área de Parques y Jardines del Ayuntamiento de Barcelona quiere modernizar su flota de mini-furgonetas con una nueva, experimental, impulsada por hidrógeno, que tiene la ventaja de ser altamente ecológica y fácil de mantener pero que tiene una autonomía muy limitada. Para resolver el problema de la autonomía se han colocado 3 surtidores de hidrógeno por diferentes partes de la ciudad para que la mini-furgoneta pueda repostar.

    Queremos planificar el recorrido diario de esta mini-furgoneta por los diferentes parques de la ciudad de forma que pueda pasar por todos los parques una vez, tardando el mínimo tiempo posible y pasando por

uno de los surtidores cada vez que se le esté acabando el hidrógeno. Como datos para esta planificación disponemos de una tabla que nos indica las posiciones de todos los parques y jardines a visitar y de los tres surtidores, así como la distancia entre cada uno de estos puntos en la ciudad. Sabemos también que la mini-furgoneta experimental tiene un tanque de n litros de hidrógeno y que su consumo es de x litros de hidrógeno por kilometro.

Comenta cada una de las posibilidades indicando si resuelven o no el problema y qué ventajas e inconvenientes tiene cada una de ellas. Justifica la respuesta.

a) Queremos utilizar A* para minimizar el número de kilometros recorridos por la furgoneta y reducir el número de repostajes al mínimo. Utilizaremos como función de coste la longitud del camino (usando la tabla de distancias antes mencionada), donde la función heurística vale infinito si la mini-furgoneta no tiene suficiente combustible para ir del punto actual a un surtidor, y en caso contrario es la suma de las distancias de los puntos por recorrer al punto actual. El operador aplicable es pasar del punto actual a otro punto.

Estamos planteando el problema como un camino, lo que es adecuado para el algoritmo A*.

El coste que asociamos al camino es la longitud recorrida. Dado que este algoritmo minimizará el coste y suponiendo que la velocidad es constante, estaremos consiguiendo el camino que tarda menos. Al minimizar el camino también se debería minimizar el número de repostajes, pues cada repostaje extra incrementará la longitud.

La función heurística no es admisible ya que la suma de las distancias del punto actual a los puntos por recorrer será superior al camino que falta. Esto hace que usarla para la búsqueda no nos garantice la solución óptima.

Encontrar una función heurística admisible es sencillo (por ejemplo se puede usar el camino mas largo entre el punto actual y los que quedan por recorrer, lo que es difícil es encontrar una función suficientemente informada para reducir suficientemente el coste computacional de la búsqueda ya que estimar el coste real del camino restante no es sencillo.

Al operador le faltaría comprobar que no se repite ningún parque, pero el usar el algoritmo A* ya nos lo garantizará ya que un camino con repetidos no será mas corto. La condición de llegar a un punto desde el que no podamos avanzar por quedarnos sin combustible queda incluida en el heurístico al hacerse infinito en esta circunstancia.

El factor de ramificación del operador es lineal respecto al número de parques.

Resumiendo, el principal problema del planteamiento es la no admisibilidad de la función heurística. La idoneidad del planteamiento se fundamenta en la posibilidad de encontrar un heurístico adecuado, sin un heurístico bueno la búsqueda con A* es demasiado costosa e intentar buscar el camino óptimo no es una buena idea.

b) Queremos utilizar busqueda local para minimizar el número de kilometros recorridos por la mini-furgoneta y reducir el número de repostajes al mínimo. Partimos de una solución inicial en la que intercalamos un paso por el surtidor de hidrógeno entre parque y parque. Se dispone de dos operadores de modificación de la solución: uno para eliminar del camino un paso por el surtidor, y otro para modificar el orden en el que visitamos alguno de los puntos del camino (ya sea parque o surtidor). La función heurística es la longitud del camino recorrido + 2 kilómetros extra de penalización por cada paso por un surtidor.

El estado inicial es una solución, es la peor que podemos encontra ya que cada vez que pasa por un parque luego pasa por una estación. Deberíamos elegir un orden de visita de los parques y con este tendriamos una solución con coste lineal en el número de parques.

Se podría encontrar la solución inicial buscando un camino con una estrategia avariciosa y un algoritmo de backtracking, usando las estaciones en el momento en el que no podamos llegar al siguiente punto del camino. Si el número de parques que podemos visitar con un único depósito es grande el coste de hallar la primera solución no debería ser demasiado grande.

El operador de eliminar un paso por el surtidor permite reducir el camino recorrido. El mayor problema

que tiene es que al ser irrevocable puede eliminar repostajes que puedan ser necesarios mas adelante para obtener mejores soluciones. Por ejemplo se pueden eliminar de la solución todos los pasos por un surtidor específico. También le falta a este operador la condición de aplicabilidad, de manera que solo se pueda eliminar un paso por el surtidor si se puede llegar del parque anterior al siguiente sin pasar por él. El factor de ramificación de este operador es el número de pasos por las estaciones de servicio que haya en la solución.

El segundo operador es correcto siempre que compruebe que el intercambio da una solución válida. El factor de ramificación de este operador será cuadrático respecto el numero de pasos del camino.

La función heurística cuanta la longitud del camino y penaliza los repostajes con un factor constante. En principio el problema solamente pide minimizar la distancia, penalizar los repostajes permitiría alejar la búsqueda de las soluciones con más repostajes, pero dado que los repostajes añaden longitud al recorrido podríamos simplemente estar añadiendo dos veces el mismo criterio. También, Dependiendo de las magnitudes de las distancias la constante de ponderación puede dar mas o menos importancia a los repostajes, por lo que habría que comprobar su efecto real.

15. El Ministerio de Sanidad y Consumo español quiere mejorar el sistema de asignación de plazas de médicos a centros hospitalarios. Cada médico m_i tiene una especialidad, acepta un sueldo mínimo determinado (Pm_i) y acepta una distancia máxima a recorrer entre su casa y su trabajo (Dm_i). Cada centro c_i dispone de x_j plazas nuevas por cada especialidad j. El ministerio paga cada plaza de especialidad j a un precio máximo determinado (P_j), y dispone de un mapa de distancias entre las viviendas de los médicos y los centros hospitalarios.

    Se quiere asignar médicos a centros de manera que se cubra el máximo número de plazas nuevas. Supondremos que el número de médicos que solicitan plaza es mucho mayor que el numero de plazas. Se nos plantean dos siguientes formas de solucionar automáticamente este problema.

    Comenta cada una de las posibilidades indicando si resuelven o no el problema, qué errores te parece que tiene cada solución y cómo se podrían corregir, y qué ventajas e inconvenientes tienen cada una de ellas. Justifica la respuesta.

    a) Queremos usar satisfacción de restricciones donde las variables son las plazas a cubrir, y sus dominios son el conjunto de médicos solicitantes m_i. Las restricciones son la especialidad, el sueldo y la distancia convenientes para los médicos y los centros.

La representación del problema como un problema de satisfacción de restricciones es correcta.

El problema pide asignar un conjunto de médicos a un conjunto de plazas, las variables son las plazas y los valores a asignar son los médicos. No se indica en el enunciado que una plaza puede no tener médico asociado, por lo que el valor vacío debería pertenecer también al dominio de las variables.

Cada variable debería tener asociado el centro al que pertenece (no se menciona en el enunciado) y su especialidad. El precio máximo es fijo para cada plaza según su especialidad (P_j).

Las restricciones que se indican son correctas: Restricción según la distancia entre la plaza y el domicilio del médico, restricción según el sueldo aceptable por el médico y la coincidencia de especialidad del médico con la de la plaza. No se indica en el enunciado que un médico solo puede aparecer asignado a una plaza.

Una representación alternativa podría ser el considerar cada tipo de plaza de un centro (segun la especialidad) como una variable y permitir asignar a una variable tantos médicos como plazas haya disponibles.

Dado que estamos planteando el problema mediante satisfacción de restricciones lo que obtendremos es una asignación válida de médicos a plazas, pero no obtendrémos la solución que maximice el número de plazas asignadas, por lo que esta aproximación no nos da la solución que se nos pide.

Una posibilidad de acercarnos a lo que se nos pide es añadir una restricción que imponga un número mínimo de plazas cubiertas, pero evidentemente haría falta aproximar el valor que pueda dar una solución.

    b) Queremos utilizar A* de manera que se pueda asignar un médico a una plaza (si las especialidades son las mismas y si el sueldo y la distancia son convenientes), y desasignar un médico de una

plaza, ambas operaciones con coste 1. La función heurística h' que se pretende usar es el numero de plazas cubiertas.

En este caso planteamos el problema como la construcción de un camino, el conjunto de asignaciones/desasignaciones de médicos a plazas.

A los operadores les faltaría comprobar que el médico no esté asignado ya. Tal como son los costes de los operadores, el operador de desasignar no aparecerá nunca en la solución ya que siempre existirá un camino alternativo en el que no se haga la asignación, por lo que es innecesario. El factor de ramificación será del número de médicos multiplicaado por el de plazas en el caso peor, aunque en la práctica será menor por las restricciones de asignación.

La función heurística no tiene demasiado sentido ya que el número de plazas cubiertas no es una estimación del número de plazas que llegaremos a cubrir y de hecho no es admisible ya que llegará un momento en el que sea superior al número de plazas por cubrir.

El número de plazas por cubrir tampoco es un buen estimador ya que por lo general será superior al número de plazas que realmente cubriremos, por lo que tampoco es admisible.

En este caso es difícil encontrar un heurístico que estime cuantas plazas de las que quedan podrán ser cubiertas que sea admisible y suficientemente cercano al coste real sin hacer la búsqueda completa, por lo que aunque el algoritmo puede obtener el óptimo el coste del obtenerlo puede ser demasiado grande.

16. Tenemos un sistema P2P que utiliza un mecanismo centralizado para asignar a cada cliente qué otros clientes son los que le envían las partes del fichero que le faltan. Cada cliente calcula una lista con los retardos medios de transmisión a cada uno de los clientes que conoce (en milisegundos). El mecanismo centralizado conoce el ancho de banda disponible de cada cliente tanto de subida como de bajada (en Kb/s) para el fichero que se quiere transmitir. Cada cierto tiempo el mecanismo centralizado distribuye a los clientes con qué otros clientes debe conectarse para recibir partes del fichero y qué ancho de banda dedicar. Para cada cliente conocemos qué partes del fichero tiene, por lo que podemos saber si puede enviar o no a un cliente. La idea es que minimicemos el tiempo de retardo total de las transmisiones y utilicemos el máximo ancho de banda de bajada disponible de cada cliente.

    Comenta cada una de las posibilidades indicando si resuelven o no el problema, qué errores te parece que tiene cada solución y cómo se podrían corregir, y qué ventajas e inconvenientes tienen cada una de ellas. Justifica la respuesta.

    a) Queremos utilizar A* de manera que recorremos la lista de clientes en un orden preestablecido. El estado es la asignación que hemos hecho de clientes y sus anchos de banda a los clientes recorridos. Utilizamos como operador asignar a un cliente uno de los que conoce (siempre que tenga partes del fichero que el cliente actual no tenga) y su máximo ancho de banda de subida al cliente actual, cuando el ancho de banda de bajada del cliente actual es superado por la suma de los anchos de banda de subida de los clientes asignados pasamos al siguiente cliente. Evidentemente una vez asignado un cliente para transmitir partes del fichero no lo podemos asignar más veces. El coste del operador es el retardo del cliente asignado. La función heurística es la suma para los clientes que quedan por recorrer de los retardos a los clientes que conocen.

Planteamos la busqueda como un camino en el que cada paso es la asignacion a un cliente a otro, correcto.

Tenemos dos operadores, uno es asignar cliente al cliente actual y otro pasar al siguiente cliente, las restricciones de los operadores hacen que el segundo operador solo se pueda aplicar cuando no se puede aplicar el primero.

En este caso el orden en que se hacen las asignaciones lo determina el coste de los operadores, que en este caso es el retardo, esto no nos da ninguna garantia de que podamos maximizar la ocupacion del ancho de banda.

La funcion heuristica nos dara valores superiores de los retardos reales que sumaran las asignaciones que hacemos, por lo que no sera admisible.

El mayor problema de esta solucion es no tener en cuenta como se asignan los anchos de banda en ningun momento.

Otro problema es no poder compartir un cliente entre varios, esto impide que se pudieran encontrar asignaciones mejores.

b) Queremos utilizar búsqueda local generando una solución inicial en la que cada cliente recibe de todos los clientes que conoce que tienen partes del fichero que le faltan con un ancho de banda de 1 Kb/s. Como operadores tenemos aumentar o disminuir el ancho de banda de un cliente que transmite a otro en 1 Kb/s. La función heurística es la suma para cada cliente de los retardos de los clientes que le transmiten con un ancho de banda superior a 0 Kb/s.

La solucion inicial que se plantea puede no ser solucion si para algun cliente el ancho de banda que soporta es inferior al numero de clientes que conoce. Tampoco tenemos en cuenta que lo que envian los clientes superen su capacidad de transmision. Por lo tanto deberiamos crear una solucion inicial que al menos respete estas restricciones.

Los operadores nos deberian permitir explorar todas las posibles asignaciones de ancho de banda entre clientes siempre que tengamos en cuenta las restricciones de subida y bajada.

La funcion heuristica solo se preocupa del retardo, por lo que explorando de esta manera la mejor solucion es en la que nadie transmite nada, que es la que menos retardo tiene. Para obtener una buena solucion deberiamos incluir restricciones que permitan maximizar el acho de banda que se recibe.

Para ello deberian penalizarse soluciones en las que el ancho de banda de un cliente no este ocupado. Una manera sencilla podria ser calcular todo el ancho de banda que necesita el colectivo de clientes y tomarlo como valor base que se podria restar del ancho de banda realmente ocupado

17. Se quiere planificar cómo componer S servicios Web en un único servicio de orden superior (meta-servicio). Cada servicio Web usa un conjunto de agentes informáticos que deben ejecutarse en un orden específico para cumplir la tarea que realiza el servicio, estos agentes pueden trabajar en paralelo. Se supone que la acción que realiza cada agente tiene la misma duración (un paso) y hay que tener en cuenta que un servicio puede necesitar un mismo agente en diferentes pasos de su ejecución. Se dispone de un agente de cada tipo, teniendo un total de A agentes. El meta-servicio se considera completo cuando se haya completado cada servicio que lo compone. Se plantean dos siguientes alternativas para minimizar el número total de pasos de ejecución del meta-servicio.

    Comenta cada una de las posibilidades indicando si resuelven o no el problema, qué errores te parece que tiene cada solución y cómo se podrían corregir, y qué ventajas e inconvenientes tienen cada una de ellas. Justifica la respuesta.

    a) Queremos utilizar A*. Definiremos el estado como la asignación de pasos de los S servicios individuales a uno de los A agentes en cada paso del meta-servicio. El estado inicial es tener un único paso del meta-servicio donde ninguno de los agentes tiene un servicio asignado.

    Los operadores de cambio de estado consisten en:

    1) Asignar el primer paso no ejecutado de alguno de los servicios a un agente libre en el paso actual del meta-servicio, con coste uno
    2) Añadir un paso nuevo al meta-servicio, con coste uno

    La función heurística es la suma de pasos de los servicios individuales que nos quedan por ejecutar dividida por el número de agentes.

Planteamos el problema como la construccion de un camino, que sera la asignacion de pasos de cada servicio del meta servicio a agentes, es el modo correcto de plantear una busqueda con A*.

Iniciar desde el estado vacio tambien es la aproximacion correcta, si planetamos la busqueda como un camino este es el estado inicial, el estado final es cuando hemos hecho la asignacion completa.

Los operadores van asignado los pasos actuales de los servicios a agentes en el paso actual del metaservicio o crean un nuevo paso del metaservicio, esto conecta los posibles estados adyacentes a uno dado. Seria mas productivo si al añadir un nuevo paso tambien asignaramos un agente a uno de sus pasos, asi evitariamos tener pasos del metasercicio vacios.

El coste de estos operadores hace que A* minimice la suma total de pasos mas la suma pasos del metaservicio.

Es un problema que el coste de añadir un nuevo paso del metaservicio y el coste de asignar un agente al paso actual sea el mismo ya que el A* explorara primero el añadir nuevos pasos, que no hacen nada, hasta que el coste supere la posibilidad de asignar un agente. Se supone que debemos meter el mayor numero de agentes en cada paso del metaservicio y con estos costes no favorecemos esa exploracion.

La exploración de A* acabara minimizándonos la suma total de pasos (que es constante) y los pasos del metaservicio (que es lo que realmente nos interesa).

El hacer que el coste del primer operador sea cero tampoco es una solucion ya que nos obligaria a explorar todas las posibles asignaciones de agente al paso actual antes de poder añadir un paso mas. Una posibilidad mejor es que el coste de añadir un nuevo paso fuera igual al numero de servicios.

La funcion heuristica dados los costes es admisible, siempre estara por debajo del coste que tienen los caminos, de hecho es una cota del numero ideal de pasos.

b) Queremos utilizar satisfacción de restricciones. Suponemos que el número máximo de pasos del meta servicio (MP) es el número de veces que aparece el agente más utilizado, de manera que usamos $S \cdot MP$ variables para representar qué agente ejecuta un paso de un servicio en la secuencia de pasos del metaservicio. El dominio de cada variable son los A agentes, mas un valor que indica que la variable no esta asignada. Las restricciones son las siguientes:

1) Para las variables de servicio en un paso, estas no pueden tener el mismo agente.
2) Para las variables de un mismo servicio, estas no pueden violar la secuencia de acciones del servicio

El problema principal de SR es que no podemos optimizar el numero de pasos porque no es lo que hace SR

Aunque no pretendiéramos optimizar el valor, tomar como numero máximo de pasos el numero de veces que aparece el agente mas utilizado no es correcto, el numero de pasos puede ser superior, por lo que no tendríamos variables suficientes para resolver el problema.

En el caso de tener variables suficiente, las restricciones que dan para el grafo de restricciones serian correctas.

Con esta aproximación corrigiendo los problemas podíamos encontrar una solución, pero no la optima

7.2 Representación

7.2.1 Problemas solucionados

10. Tenemos tres frames A, B, C y las definiciones de las siguientes relaciones:

	R_{AB}	R_{BC}	R_{CC}
Dominio	A	B	C
Rango	B	C	C
Cardinalidad	N	1	1
Inversa	R_{AB}^{-1} (1)	R_{BC}^{-1} (N)	R_{CC}^{-1} (1)
Transitiva	no	no	no

1) ¿Podría ser la relación R_{CC} transitiva? Justifica tu respuesta.

Imposible, es una relación 1 a 1, por lo que no podrían existir la relaciones entre instancias que formaran la transitividad.

2) ¿Es posible Definir un slot *sb* en el frame B con un demon `if-needed` con parámetro x que retorne todas las instancias de C que tengan el valor de x en el slot *sc* (*sc* y x son del mismo tipo)? Impleméntalo si la respuesta es afirmativa o justifica tu respuesta si es negativa.

Obviamente no, no se puede implementar, un demon no puede tener parámetros.

3) Define un método en el frame B que retorne todas las parejas de instancias de B que tengan una relación R_{AB}^{-1} con la misma instancia de A y tal que ambas tengan ademas una relación R_{BC} con alguna instancia de C.

Function: metodoB()
linsB=B.tiene-por-instancias()
lparB=∅
for *ib1* ∈ *linsB* **do**
  for *ib2* ∈ *linsB* **do**
    if *(ib1.*R_{AB}^{-1} *== ib2.*R_{AB}^{-1}*)* **then**
      if *(card?(ib1.*R_{BC}*)≠0 ∧ card?(ib2.*R_{BC}*)≠0)* **then**
        lparB=lparB ∪ {ib1,ib2}
      end
    end
  end
end
return *lparB*

4) Queremos definir el slot *sa* en el frame A de tipo booleano y cardinalidad uno y una relación entre los frames A y C de manera que las instancias de C puedan heredar este slot. Define el slot y la relación de manera que esta herencia sea posible suponiendo que la cardinalidad del slot no se puede cambiar (si hay varias posibilidades indícalo).

Slot	sb
Dominio	A,C
Rango	Booleano
Cardinalidad	1
Valor	—
Demons	—
Herencia	Tax=si, Usu=si

Relación	R_{AC}
Dominio	A
Rango	C
Cardinalidad	N ∗
Inversa	R_{CA}^{-1} (1)
Compuesta	—
Transitiva	—
Demons	—
Herencia	sb

∗ La cardinalidad de la relación R_{AC} puede ser 1 o N.
La limitación viene porque dado que la cardinalidad del slot es 1 y es de tipo booleano no tiene sentido que en C se herede de mas de una instancia, por lo que la cardinalidad de la inversa solo puede ser 1.