

# Algorismes Bàsics per la Intel·ligència Artificial

## Col·lecció de Problemes

Departament de Ciències de la Computació

Grau en Intel·ligència Artificial

Curs 2024/2025 1Q



**FIB**

Facultat d'Informàtica  
de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA



Copyleft  2022-2024

DEPARTAMENT DE CIÉNCIES DE LA COMPUTACIÓ

FACULTAT D'INFORMÀTICA DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Licensed under the Creative Commons Attribution-NonCommercial 3.0 Unported License (the “License”). You may not use this file except in compliance with the License. You may obtain a copy of the License at <http://creativecommons.org/licenses/by-nc/3.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Imágenes generadas con Stable Diffusion

*Primera edición, septiembre 2022*

*Esta edición, septiembre 2024*

En l'elaboració de la col.lecció de problemes d'IA han participat els professors:

Sergio Álvarez Napagao

Javier Béjar Alonso

Victor Giménez Ábalos

Jordi Turmo Borrás

Javier Vázquez Salceda

Responsable de la publicació: Javier Béjar ([bejar@cs.upc.edu](mailto:bejar@cs.upc.edu))



## Índice general

I	Cerca	
1	Representació de Problemes de Cerca .....	1
2	Cerca Heurística .....	5
3	Cerca Local .....	15
4	Jocs .....	33
5	Satisfacció de restriccions .....	43
6	Anàlisi de mètodes de cerca .....	53
II	Planificació	
7	Planificació .....	71





# Cerca

<b>1</b>	<b>Representació de Problemes de Cerca . . . . .</b>	<b>1</b>
<b>2</b>	<b>Cerca Heurística . . . . .</b>	<b>5</b>
<b>3</b>	<b>Cerca Local . . . . .</b>	<b>15</b>
<b>4</b>	<b>Jocs . . . . .</b>	<b>33</b>
<b>5</b>	<b>Satisfacció de restriccions . . . . .</b>	<b>43</b>
<b>6</b>	<b>Anàlisi de mètodes de cerca . . . . .</b>	<b>53</b>





## 1. Representació de Problemes de Cerca

1. Tenemos un tablero de  $3 \times 3$  casillas como el de la figura

N		N
B		B

En cada esquina tenemos un caballo de ajedrez, dos caballos negros y dos blancos. Deseamos intercambiar los caballos negros con los blancos.

- (a) Define que elementos forman el estado, el estado inicial y cuál es estado final o que propiedades ha de cumplir. Estima el tamaño del espacio de estados.  
(b) Define las características de los operadores para realizar la búsqueda (condiciones de aplicabilidad y función de transformación), evalúa cual sería el factor de ramificación (aproximadamente).  
(c) ¿Importa el camino o solo el estado final? ¿La solución ha de ser óptima?
2. Dispones de dos jarras de agua, una de 4 litros y otra de 3 litros. Tiene un grifo que te permite llenar totalmente las jarras de agua, necesitas obtener exactamente 2 litros en la jarra de cuatro litros  
(a) Define que elementos forman el estado, el estado inicial y cuál es estado final o que propiedades ha de cumplir. Estima el tamaño del espacio de estados.  
(b) Define las características de los operadores para realizar la búsqueda (condiciones de aplicabilidad y función de transformación), evalúa cuál sería el factor de ramificación (aproximadamente).  
(c) ¿Importa el camino o sólo el estado final? ¿La solución ha de ser óptima?
3. En el procedimiento de validación por resolución en lógica de enunciados, se puede validar cualquier razonamiento a partir de la transformación de sus premisas y la negación de la conclusión a forma normal conjuntiva. El algoritmo más sencillo (pero no muy eficiente) de validar un razonamiento es aplicar la regla de la resolución ( $P \vee Q, \neg P \vee R \therefore Q \vee R$ ) sistemáticamente entre las cláusulas hasta conseguir derivar una contradicción.  
(a) Define que elementos forman el estado, el estado inicial y cual es estado final o que propiedades ha de cumplir. Estima el tamaño del espacio de estados.

- (b) Define las características de los operadores para realizar la búsqueda (condiciones de aplicabilidad y función de transformación), evalúa cual sería el factor de ramificación (aproximadamente).
- (c) ¿Importa el camino o solo el estado final? ¿La solución ha de ser óptima?
- (d) Piensa en el mecanismo de resolución lineal y vuelve a responder a los apartados a y b
4. Si piensas en el juego del tetris, este consiste en cubrir la máxima área de un rectángulo de  $N \times M$  sin dejar huecos (o dejando los mínimos posibles) utilizando un conjunto ordenado de piezas con todas las posibles formas construibles utilizando cuatro cuadrados (siete piezas distintas)
- (a) Define que elementos forman el estado, el estado inicial y cuál es estado final o que propiedades ha de cumplir. Estima el tamaño del espacio de estados.
- (b) Define las características de los operadores para realizar la búsqueda (condiciones de aplicabilidad y función de transformación), evalúa cuál sería el factor de ramificación (aproximadamente).
- (c) ¿Importa el camino o solo el estado final? ¿La solución ha de ser óptima?
5. Continuando con el juego del tetris este utiliza como piezas todas las formas posibles usando cuatro cuadrados de manera que cada cuadrado tenga al menos un lado contiguo a otro cuadrado. Supón que quieres calcular todas las formas posibles que se pueden construir con N cuadrados.
- (a) Plantéalo como un problema de búsqueda en espacio de estados definiendo los estados y los operadores necesarios para realizar la búsqueda (evidentemente no hay una única manera de plantear el problema, define las que se te ocurran y evalúa sus diferencias)
6. Dada una permutación de números de 1 a n queremos ordenarla utilizando el menor número de operaciones de inversión de un intervalo, donde una inversión de intervalo se define a partir de un par de posiciones (i,j) y su efecto es invertir el orden en el que están todos los números entre esas posiciones.
- (a) Define que elementos forman el estado, el estado inicial y cuál es estado final o que propiedades ha de cumplir. Estima el tamaño del espacio de estados.
- (b) Define las características de los operadores para realizar la búsqueda (condiciones de aplicabilidad y función de transformación), evalúa cuál sería el factor de ramificación (aproximadamente).
7. Existen múltiples problemas clásicos sobre grafos que se pueden plantear como una búsqueda en espacio de estados. La gracia de estos problemas es que muchos problemas reales se pueden transformar a estos. Intenta plantear los siguientes:

El viajante de comercio: Un viajante de comercio desea visitar un conjunto de ciudades partiendo de una dada y acabando en esta, sin repetir ninguna y recorriendo el mínimo de distancia (se supone que dispone de un mapa que indica las conexiones entre ciudades y sus distancias)

El k-viajante de comercio: Ahora el viajante quiere obtener k caminos de mínima longitud que comiencen y terminen en una ciudad y que en los k caminos cada ciudad aparezca como mínimo en un camino (es decir, entre los k caminos recorremos todas las ciudades)

El Cartero chino: Un cartero (chino) desea poder repartir el correo por su zona en el mínimo tiempo posible, para ello necesita obtener un recorrido que pase por todas las calles al menos una vez (se supone que tenemos el mapa de todas las calles de la zona) (también se puede plantear la misma variante de k caminos)

Máximo Clique: Un clique es un grafo en el que cada vértice está conectado con el resto de vértices del grafo (o sea que es un grafo completo). El problema consiste en encontrar para un grafo el mayor subgrafo que sea un clique.

Mínimo k-árbol de expansión mínima: Dado un grafo en el que cada arista tiene un peso se trata de encontrar el subgrafo sin ciclos (árbol) que contenga k aristas y que tenga el mínimo peso.

Mínimo árbol de Steiner: Dado un grafo completo (todos los vértices conectados con todos), donde cada arco tiene un peso, y un subconjunto de vértices del grafo, obtener el grafo de coste mínimo

---

que contenga el conjunto de vértices (el grafo puede contener más vértices, la gracia es conectar unos vértices específicos con el mínimo coste)

Coloreado de grafos: Dado un grafo y k colores, asignar a cada vértice del grafo un color de manera que dos vértices conectados no tengan el mismo color.

- (a) Define que elementos forman el estado, el estado inicial y cuál es estado final o que propiedades ha de cumplir, estima el tamaño del espacio de estados. Piensa que puede haber diferentes formas de plantear el problema.
- (b) Define las características de los operadores para realizar la búsqueda (condiciones de aplicabilidad y función de transformación), evalúa cuál sería el factor de ramificación (aproximadamente).

8. Otros problemas típicos que se pueden plantear como búsqueda en espacio de estados son los que involucran horarios, piensa en los siguientes problemas:

Tienes que distribuir un conjunto de cursos de diferentes niveles (m cursos por cada nivel, n niveles) en un horario semanal (5 días) en un aula sabiendo que todos los cursos son de una hora, pero que no quieres que haya más de k cursos de un nivel cada día. Generaliza el problema suponiendo que tienes g grupos de cada curso y a aulas y no quieres que haya dos cursos iguales a la misma hora. Supón que tienes que hacer tu horario cuatrimestral y dispones de m asignaturas, cada una de ellas con n grupos y quieras elegir un subconjunto k de ellas ( $k < m$ ) sin que se solapen sus horarios

- (a) Define que elementos forman el estado, el estado inicial y cuál es estado final o que propiedades ha de cumplir, estima el tamaño del espacio de estados. Piensa que puede haber diferentes formas de plantear el problema.
- (b) Define las características de los operadores para realizar la búsqueda (condiciones de aplicabilidad y función de transformación), evalúa cuál sería el factor de ramificación (aproximadamente).

9. El problema del club de golf. Tenemos un club de golf con 32 jugadores, todos ellos juegan una vez a la semana en grupos de 4 personas. Queremos obtener los emparejamientos para el mayor número de semanas de manera que un jugador no este en el mismo grupo que otro jugador mas de una vez.

- (a) Define que elementos forman el estado, el estado inicial y cuál es estado final o que propiedades ha de cumplir, estima el tamaño del espacio de estados. Piensa que puede haber diferentes formas de plantear el problema.
- (b) Define las características de los operadores para realizar la búsqueda (condiciones de aplicabilidad y función de transformación), evalúa cuál sería el factor de ramificación (aproximadamente).

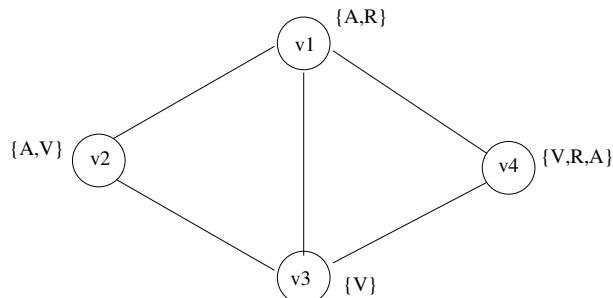




## 2. Cerca Heurística

1. El problema del coloreado de grafos consiste en etiquetar con un color cada vértice del grafo, de forma que no haya dos vértices adyacentes con el mismo color. Este problema se puede resolver por medio de búsqueda.

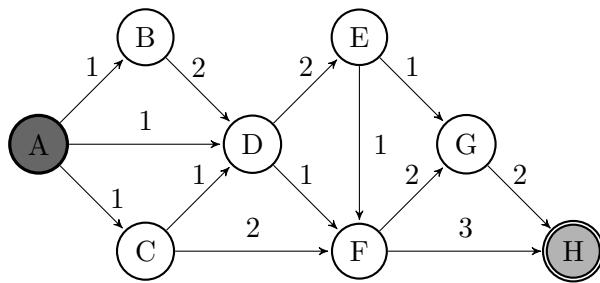
(a) Describe los pasos que seguiría un algoritmo A\* para encontrar una solución al coloreado del siguiente grafo:



en donde los vértices son V1, V2, V3 y V4 y los colores posibles para cada vértice se indican entre llaves (A:azul, R:rojo, V:verde). El estado inicial tiene todos los vértices sin color asignado. Un estado S' es sucesor de otro S si S' contiene la asignación de colores de S más la asignación de un color al vértice más pequeño sin colorear de S (es decir, los vértices se instancian en el orden V1 → V4). La función heurística es:

$h(s) = \text{número de vértices sin color en } s + 2 \times \text{número de vértices adyacentes con el mismo color en } s$

- (b) ¿Es admisible la función heurística? Razona la respuesta.  
(c) Sugiere otro algoritmo que solucione este problema de forma más eficiente que A\*
2. 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 H. 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
h(nodo)	4	4	3	3	2	3	2	0

3. Aplicar l'algoritme A\* per a trobar un camí solució del graf descrit a la taula:

	b	c	d	e	h
a	10	4			3
b			10		4
c			12	2	12
d					0
e	2		4		4

on els nodes de les files són els nodes-pares i els nodes de les columnes els nodes-fills. El node a és l'inicial i el final és d. Cada casella de la taula indica el cost de l'arc que uneix els dos nodes corresponents. La darrera columna indica el valor de la funció heurística (h).

- (a) Mostrar l'arbre de cerca generat per l'algoritme, indicant l'ordre d'expansió dels nodes, reopertures de nodes tancats, etc. Quin és el camí trobat? Quin és el seu cost? És l'òptim?
  - (b) Es verifiquen les condicions d'admissibilitat? Justifiqueu la resposta.
  - (c) Fer el mateix amb l'algoritme IDA\*.
4. Disponemos de un casillero con cuatro monedas colocadas de la siguiente forma:



El anverso de la moneda está representado por A y el reverso por R. Son posibles los siguientes movimientos:

- Desplazamiento (coste=1): Una moneda puede ser desplazada a la casilla contigua si esta se encuentra vacía.
- Giro (coste=1): Cualquier moneda puede ser girada sin ninguna condición adicional. Solo una cada vez.
- Salto (coste=2): Una moneda puede saltar sobre su vecina si a continuación hay una casilla vacía, es decir, solo es posible saltar por encima de una moneda. Cuando una moneda salta, cae realizando un giro. Un ejemplo de salto (coste=2) es pasar del estado AR\_RA al estado ARRRA\_

Deseamos obtener la situación final siguiente:



Dada la función heurística  $h(n) = p2 + p3 + p4 + p5 + dv$

donde  $p_i$  vale 0 si la casilla  $i$  contiene la asignación correcta respecto del estado final y vale 1 en caso contrario y  $dv$  es la distancia del blanco respecto a la posición final (casilla 1).

Por ejemplo,  $h(\text{estado inicial}) = 1 + 4 = 5$

- (a) Resolver el problema aplicando A\*. Indica claramente el orden de expansión de los nodos, el tratamiento de nodos duplicados, los valores de las funciones, el camino obtenido y su coste.
- (b) El camino obtenido es de coste óptimo? El heurístico usado es admissible? Por qué?
- (c) Resolver el problema aplicando IDA\*. Indica claramente el proceso en cada nivel: orden de expansión de los nodos, nodos duplicados... Se obtiene un camino de coste óptimo?

5. Tenemos cinco monedas dispuestas de la siguiente forma:

A R A R A

El anverso de la moneda está representado por A y el reverso por R. Se considera un movimiento (de coste 1) el dar la vuelta a dos monedas contiguas.

Deseamos obtener la situación final siguiente:

R R R A R

Dada la función heurística  $h(n)$  = número de monedas mal colocadas,

- (a) Resolver el problema aplicando A\*. Indica claramente el orden de expansión de los nodos, el tratamiento de nodos duplicados, los valores de las funciones, el camino obtenido y su coste.
- (b) ¿El camino obtenido es de coste óptimo? ¿El heurístico usado es admissible? ¿Por qué?
- (c) Resolver el problema aplicando IDA\*. Indica claramente el proceso en cada nivel: orden de expansión de los nodos, nodos duplicados ¿Se obtiene un camino de coste óptimo?

6. Supón que tenemos un computador en el que varias instrucciones se pueden ejecutar a la vez. La unidad que ejecuta las instrucciones puede ejecutar en paralelo instrucciones de tres tipos: A, B y C. La instrucción A tarda 3 ciclos de reloj, la instrucción B tarda 2 ciclos y la instrucción C tarda 1 ciclo. Esta unidad es capaz de ejecutar en un paso tantas instrucciones como quepan dentro de una ventana de 3 ciclos de reloj, de manera que se puede utilizar para ejecutar a la vez varios procesos (hilos de ejecución). Por ejemplo, si tenemos estos 3 hilos de instrucciones H1(ABC) H2(BCA) H3(CCC) como estado inicial, la unidad podría ejecutar las instrucciones de la manera que aparecen en la tabla:

	A	B	C
1 <sup>er</sup> paso	H1	H2	H3
	H1	H2	H3
	H1		H2

Se ejecuta la instrucción A del primer hilo en paralelo con la B del segundo hilo, las dos primeras C del tercer hilo y finalmente la C del segundo hilo respetando así la precedencia de instrucciones del segundo hilo.

Estado actual: H1(BC), H2(A), H3(C)

	A	B	C
2 <sup>o</sup> paso	H2	H1	H3
	H2	H1	
	H2		H1

Se ejecuta la instrucción A del segundo hilo en paralelo con la B del primer hilo, la tercera C del tercer hilo y finalmente la C del primer hilo respetando así la precedencia de instrucciones del primer hilo.

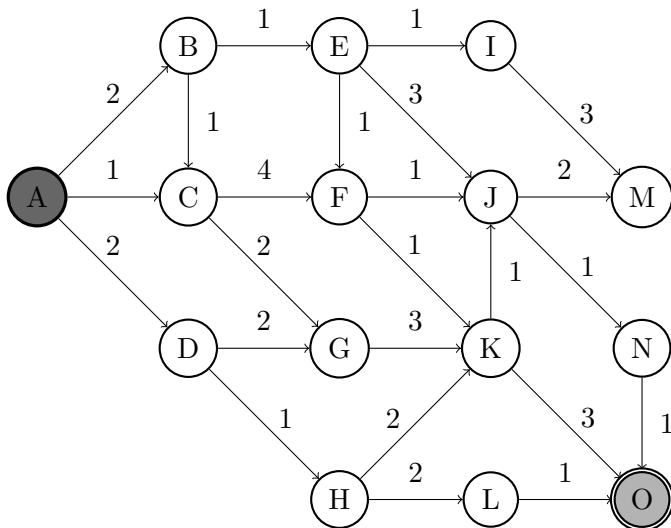
La secuencia de ejecución sería: 1er paso : [(A,H1),(B,H2),(C,H3),(C,H3),(C,H2)],  
2o paso: [(A,H2),(B,H1),(C,H3),(C,H2)]

La idea es secuenciar varios hilos en paralelo en tiempo de compilación, de manera que el número de pasos de ejecución sea el mínimo. La búsqueda consiste en determinar qué conjunto de instrucciones y de qué hilos se pueden ejecutar en cada paso de la unidad de ejecución. En caso de que se pueda elegir entre varias instrucciones iguales en un paso, el orden de exploración se seguirá escogiendo la instrucción según el orden de los hilos.

Los hilos a compilar serán: H1(ABAC) H2 (CABA) H3(ACBA)

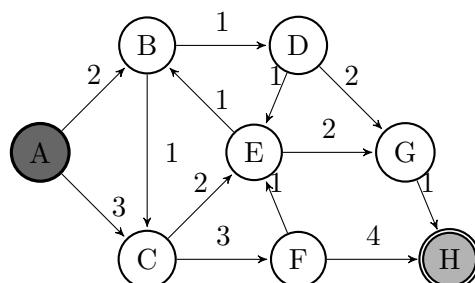
- (a) Utiliza el algoritmo del A\* para encontrar la secuencia de menos pasos para estos hilos de ejecución (indica claramente el orden de expansión de los nodos). Considera que el coste de cada paso es 1. Como heurístico para la búsqueda utiliza el siguiente:  
 $h_1(n)$ = número de instrucciones del hilo con más instrucciones pendientes
- (b) El heurístico no es admisible ¿Por qué? ¿La solución es óptima?
- (c) Supón el heurístico:  
 $h_2(n)$ = número de instrucciones A que quedan por ejecutar  
¿Sería admisible este heurístico?
- (d) Utiliza ahora el algoritmo de IDA\* con este nuevo heurístico para hacer la exploración (realiza cada iteración por separado indicando el orden de expansión de los nodos)

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 repeticiones de los nodos y los cambios de coste que aparezcan. ¿Es la función heurística admisible?



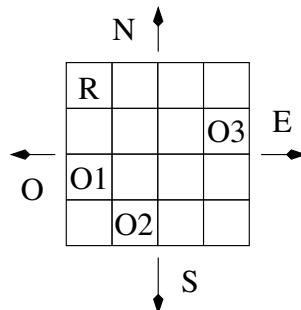
Nodo	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$h(nodo)$	6	5	6	6	3	5	5	4	8	3	2	1	5	1	0

8. Dado el siguiente grafo, donde cada arco indica su coste, y la tabla que indica la estimación del coste  $h$  hasta la solución, indica cuál 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 H. 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
$h(nodo)$	6	4	3	3	2	3	2	0

9. Deseamos hallar la ruta que ha de seguir un robot para recoger un conjunto de objetos en un orden específico en una habitación. La situación es la que se presenta en la figura. El robot sólo se puede desplazar en vertical y en horizontal. En cada movimiento sólo se desplaza una casilla. Cada vez que el robot llega a la casilla donde hay un objeto lo recoge y lo lleva consigo.



- (a) Utiliza el algoritmo del A\* para hallar el camino que permite al robot recoger los 3 objetos en orden usando la función heurística:

$$h_1(n) = \sum d_{rj}$$

Donde  $d_{rj}$  es la distancia en movimientos del robot al objeto  $j$ . Considera que, cuando el robot recoge un objeto, su distancia pasa a ser 0 y que el objeto se mueve con el robot una vez recogido. Para hacer la expansión de los nodos utiliza el orden N-S-E-O tal como se indica en la figura. Indica el orden de expansión de los nodos. ¿El camino encontrado es óptimo?

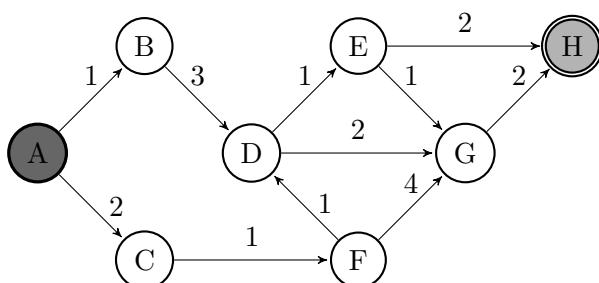
- (b) Supón que el heurístico es:

$$h_2(n) = d_{r1} + d_{12} + d_{23}$$

Donde  $d_{r1}$  es la distancia en movimientos del robot al objeto O1,  $d_{12}$  es la distancia en pasos del objeto O1 al objeto O2 y  $d_{23}$  es la distancia del objeto O2 al objeto O3. Utiliza de nuevo el algoritmo del A\* para hallar la solución. Para hacer la expansión de los nodos utiliza el orden N-S-E-O tal como se indica en la figura. Indica el orden de expansión de los nodos. ¿El camino encontrado es óptimo?

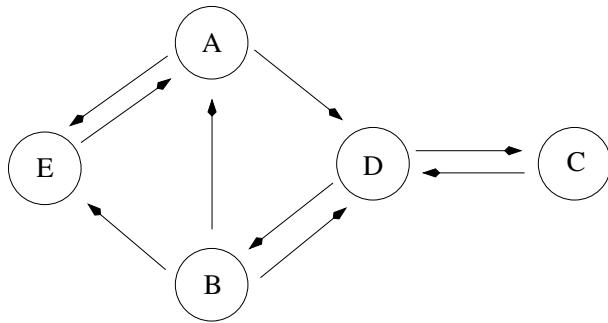
- (c) ¿Son admisibles los heurísticos usados? Podemos afirmar que uno es más informado que el otro? Razona las respuestas.
- (d) Haz de nuevo la búsqueda usando el algoritmo de IDA\* y el heurístico  $h_1$ . Indica el orden de expansión de los nodos. ¿El camino encontrado es óptimo?

10. 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 H. 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
$h(\text{nodo})$	4	4	3	3	2	3	2	0

11. Dado el siguiente grafo



- (a) Buscar, aplicando A\*, un recorrido que pase por todos los nodos empezando y acabando por el nodo A. Solo es posible pasar de un nodo X a un nodo Y si existe el arco X → Y. Se permite pasar más de una vez por el mismo nodo. En cada iteración indica el orden de expansión de los nodos.

Utiliza como función heurística el número de nodos nuevos pendientes de visitar. Orden de generación de sucesores: orden lexicográfico.

- (b) ¿El camino encontrado es óptimo? Justifica formalmente la respuesta.  
 (c) Repite la búsqueda utilizando el algoritmo IDA\*

12. Tenemos el siguiente rompecabezas con la configuración de la figura

N	N	N	B	B	B
---	---	---	---	---	---

y disponemos de una única regla, dos fichas se pueden intercambiar si están adyacentes, considerando que los dos extremos son adyacentes (es un tablero circular).

El objetivo es obtener la siguiente configuración:

B	B	N	B	N	N
---	---	---	---	---	---

- (a) Aplica el algoritmo A\* para resolver el puzzle usando la siguiente función heurística  $h(n)$  = número de posiciones diferentes entre el estado actual y la configuración final.  
 ORDEN: Al expandir los nodos considera los movimientos secuencialmente, empezando por el de más a la izquierda.

Indica el orden de expansión de los nodos y la gestión de duplicados.

- (b) Resuelve el mismo problema aplicando el algoritmo de IDA\*. Indica el orden de expansión de los nodos y la gestión de duplicados.  
 (c) ¿Es admisible el heurístico? ¿Por qué?

13. Un problema en genética es averiguar si dos cadenas de ADN son similares o no. Para ello se intentan alinear los símbolos de las secuencias y asignar una puntuación al alineamiento. El problema es que hay muchas maneras de alinear dos secuencias y solo interesa la mejor.

Para simplificar el problema, vamos a suponer que tenemos secuencias con solamente dos símbolos: A y B. El alineamiento de dos secuencias se hace símbolo a símbolo de izquierda a derecha. Como es posible que las dos secuencias sean de longitud diferente, se pueden introducir símbolos blancos durante el alineamiento para compensarlo, de manera que al final del alineamiento las dos cadenas tengan la misma longitud. Podemos haber introducido símbolos blancos en ambas cadenas.

Tenemos dos operaciones:

- 1) Crear un emparejamiento entre dos símbolos de la secuencia. El coste de esta operación es cero si los dos símbolos coinciden y uno, si no coinciden.

- 2) Añadir un blanco en una secuencia, de manera que avanzamos un símbolo en una de las secuencias y la otra la alargamos con este blanco. El coste de esta operación es dos.

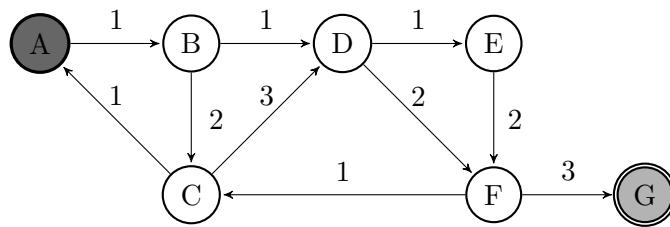
Por ejemplo, si tenemos las secuencias AABB y BAA, podríamos alinear los dos primeros símbolos, de manera que estaríamos en el estado (A|ABB, B|AA) con coste uno ya que los símbolos no coinciden (la barra vertical | indica hasta donde llega el alineamiento que llevamos), o podríamos añadir un hueco en la primera secuencia, de manera que estaríamos en el estado (\_|AABB, B|AA), con coste dos.

- (a) Aplicar el algoritmo A\* para alinear las secuencias AB y BAAB , usando como función heurística  $h(n) = |\text{longitud del trozo de la secuencia 1 por alinear} - \text{longitud del trozo de la secuencia 2 por alinear}|$

Al expandir los nodos, en el caso de añadir blancos, primero se añadirá a la primera secuencia y después a la segunda. Indica el orden de expansión de los nodos y la gestión de duplicados, si procede.

- (b) Resuelve el mismo problema aplicando el algoritmo del IDA\*. Indica el orden de expansión de los nodos y la gestión de duplicados, si procede.  
(c) ¿Es admisible el heurístico? ¿Por qué?

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
$h(\text{nodo})$	3	4	3	4	3	3	0

15. Dada la siguiente configuración inicial del tablero:

N	N	B	B	
---	---	---	---	--

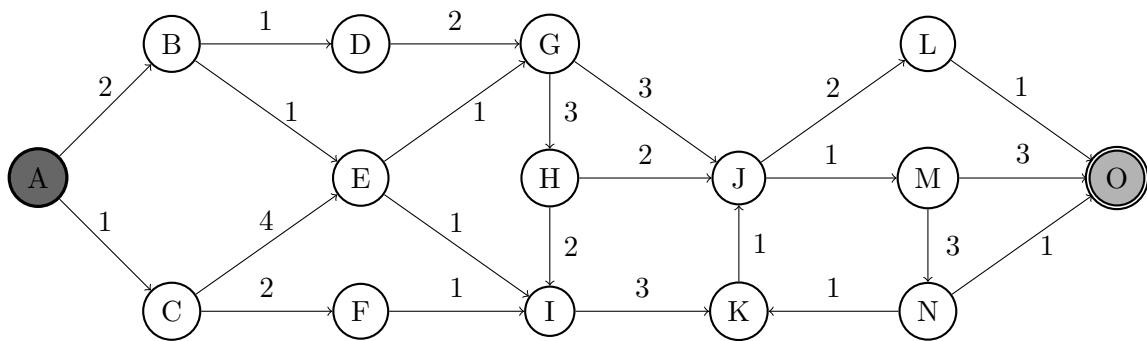
donde hay dos piezas negras (N), dos blancas (B) y una posición vacía, y teniendo en cuenta las siguientes reglas:

- (i) Una pieza puede moverse a la posición adyacente vacía con coste 1.
- (ii) Una pieza puede saltar por encima de otra (solo una) para colocarse en la posición vacía con coste 1. Si la pieza que salta es de distinto color que la saltada, esta última cambia de color.

Queremos conseguir que todas las piezas del tablero sean negras. La posición final de la casilla vacía es indiferente.

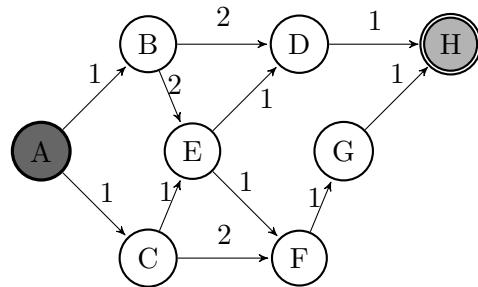
- (a) Muestra el árbol de búsqueda generado por el algoritmo A\* indicando el orden de expansión de los nodos, el tratamiento de los duplicados, la solución encontrada y el coste de la misma. Utiliza como heurístico:  $h(n) = \text{número de piezas blancas}$
- (b) Justifica la admisibilidad o no del heurístico utilizado.

- (c) Considera ahora el siguiente heurístico:  $h_2(n) = \text{número de piezas blancas} - c$  donde  $c$  vale 1 si el espacio es adyacente a una pieza blanca y vale 0 en caso contrario. Justifica su admisibilidad o no.
- (d) Razona si  $h_2(n)$  es más informado que  $h(n)$ .
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
$h(\text{nodo})$	6	5	6	6	3	5	5	4	8	3	2	1	5	1	0

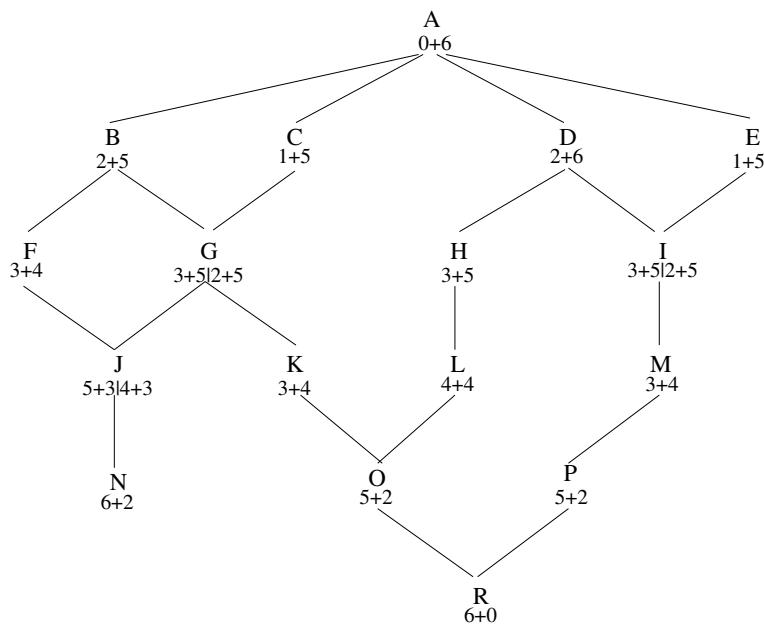
17. Dado el siguiente grafo, donde cada arco indica su coste, y la tabla que indica la estimación del coste  $h$  hasta la solución, indica cuál sería el árbol de búsqueda que se obtendría mediante el algoritmo de A\* e IDA\* para encontrar el camino entre el nodo A y el nodo H. Haz la generación de 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
$h(\text{nodo})$	6	4	4	4	4	3	3	0

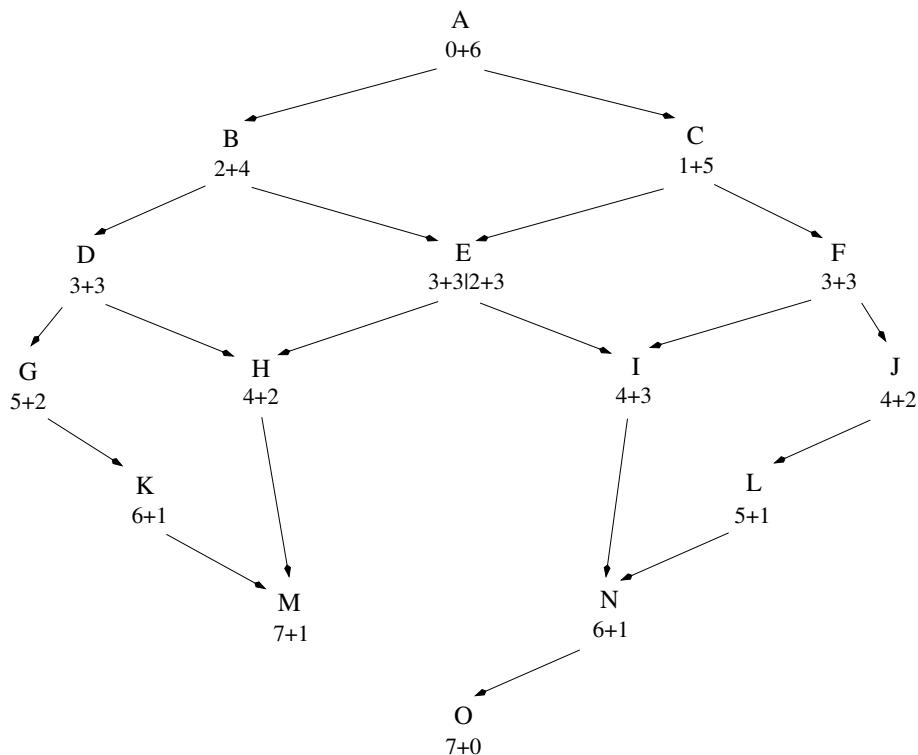
18. Dado el árbol de búsqueda que aparece en la figura indica cual sería el recorrido que haría el algoritmo A\* e IDA\*. ¿la función heurística es admisible?

En el árbol la función heurística está descompuesta en la g y la h. Cuando en un nodo aparecen dos valores el orden corresponde a la rama por la que se ha llegado al nodo.



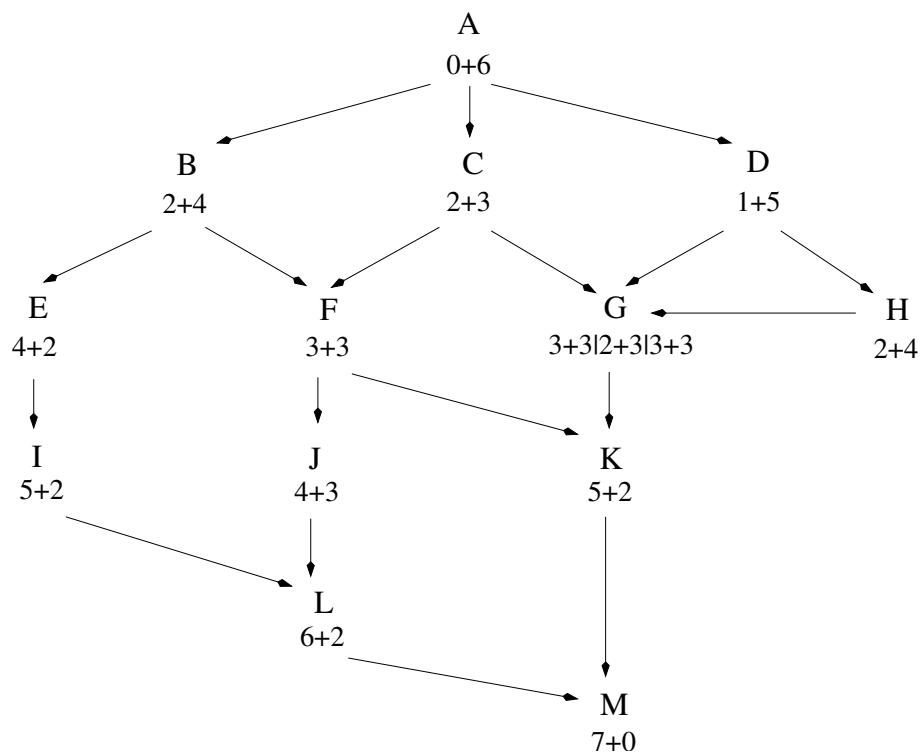
19. Dado el árbol de búsqueda que aparece en la figura indica cuál sería el recorrido que haría el algoritmo A\* e IDA\*. ¿la función heurística es admisible?

En el árbol la función heurística está descompuesta en la g y la h. Cuando en un nodo aparecen dos valores el orden corresponde a la rama por la que se ha llegado al nodo.



20. Dado el árbol de búsqueda que aparece en la figura indica cuál sería el recorrido que haría el algoritmo A\* e IDA\*. ¿la función heurística es admisible?

En el árbol la función heurística está descompuesta en la g y la h. Cuando en un nodo aparecen dos valores el orden corresponde a la rama por la que se ha llegado al nodo.





### 3. Cerca Local

1. Existen problemas en los que dado el tamaño de su espacio de búsqueda es imposible plantearse encontrar la solución óptima. Para este tipo de problemas se han diseñado algoritmos de búsqueda local (hill-climbing, simulated annealing, algoritmos genéticos)

Estos son ejemplos de este tipo de problemas:

- (a) El viajante de comercio:

Un viajante de comercio desea visitar un conjunto de ciudades partiendo de una dada y acabando en ésta, sin repetir ninguna y recorriendo el mínimo de distancia (se supone que se dispone de un mapa que indica las conexiones entre ciudades y sus distancias)

- (b) Mínimo k-árbol de expansión mínima:

Dado un grafo en el que cada arista tiene un peso, se trata de encontrar el subgrafo sin ciclos (árbol) que contenga  $k$  aristas y que tenga el mínimo peso

- (c) Mínimo árbol de Steiner:

Dado un grafo completo (todos los vértices están conectados con todos), donde cada arco tiene un peso, y un subconjunto de vértices del grafo, obtener el grafo de coste mínimo que contenga el conjunto de vértices (el grafo puede contener mas vértices, la gracia es conectar unos vértices específicos con el mínimo coste)

- (d) Bin packing en una dimensión:

Disponemos de un conjunto de contenedores y un conjunto de paquetes que colocar en ellos. Los contenedores son todos iguales y se pueden llenar hasta cierta altura. Los paquetes tienen las mismas dimensiones en su base que los contenedores, pero diferentes alturas, siempre por debajo de la altura de los contenedores. El problema consiste en: dado un conjunto de paquetes saber cual es el mínimo conjunto de contenedores necesario para transportarlos.

- (e) Knapsack problem:

También conocido como el problema de la mochila. Dado un conjunto de objetos con cierto volumen y cierto valor y un contenedor con una capacidad fija, hallar el conjunto de objetos que nos debemos llevar para maximizar la ganancia.

- (f) Weighted max-cut:

Dado un grafo con pesos en los arcos, partirlo en dos grafos de manera que se maximice la suma total de los arcos de los dos subgrafos

Responde para cada problema los siguientes apartados:

- Determina que elementos forman el estado y estima el tamaño del espacio de búsqueda
- Dado que los algoritmos que se pueden utilizar suelen partir de una solución, busca métodos con los que se puede encontrar la solución de partida (no tiene por qué ser buena)
- Define lo necesario para solucionar el problema mediante el algoritmo de hill-climbing (función heurística, operadores de cambio de estado)
- Define lo necesario para solucionar el problema mediante algoritmos genéticos (representación del estado, función de calidad, operadores de cruce y mutación)

2. Los propietarios del hotel “Bienestar” trabajan habitualmente con  $N$  tour-operadores para los cuales deben reservar siempre un conjunto de habitaciones. A principios de año, cada operador realiza su petición de habitaciones para todo el año. Las peticiones son consideradas como el número máximo a reservar para cada tour-operador, pero los propietarios saben que no tienen habitaciones suficientes para cubrir todas las peticiones de todos los tour-operadores. Por esta razón, internamente tienen definido un número mínimo a asignar a cada tour-operador. Este número les permite mantener las buenas relaciones. A partir de aquí asignarán a los tour-operadores un número de habitaciones que estará entre el mínimo interno y la petición real recibida.

Para asignar las habitaciones, los propietarios tienen varias restricciones a respetar. Por un lado, hay un número mínimo de habitaciones que queda siempre bajo la gestión directa del hotel. Por otro lado, se quiere maximizar los beneficios obtenidos de las habitaciones reservadas, teniendo en cuenta que los beneficios  $B_i$  son distintos dependiendo del tour-operador y el beneficio por habitación gestionada por el hotel es  $B_H$ . Adicionalmente, se quiere maximizar la “calidad de la ocupación” para lo cual tienen asignado a cada tour-operador un índice  $Q_i$  que cuantifica la calidad de los turistas que suelen venir a través del operador en cuestión. Este índice se asociará a cada habitación reservada para ese operador. El propio hotel tiene también un índice  $Q_H$ .

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone respecto a si es correcta, es eficiente, o es mejor o peor respecto a otras alternativas posibles. Justifica tu respuesta.

- (a) Se plantea aplicar Hill-climbing usando como solución inicial asignar el mínimo de habitaciones a cada tour-operador. Se define como operador asignar una habitación a alguien distinto del que la tiene asignada (tour-operador/hotel), siempre y cuando el nuevo tour-operador tenga mejor índice  $Q$  y se respeten todos los mínimos. Se plantea como función de evaluación la suma total de los beneficios obtenidos de cada habitación del hotel.
- (b) Se plantea aplicar Hill-climbing usando como solución inicial asignar el mínimo de habitaciones a cada tour-operador y del conjunto de habitaciones no asignadas apartar el mínimo para el hotel y asignar el resto al tour-operador que nos da mayor beneficio. Se define como operador asignar una habitación a un tour-operador distinto del que la tiene asignada. Se plantea como función de evaluación la suma total de los beneficios obtenidos por habitación más la suma del índice  $Q$  de cada una de ellas.
- (c) Se plantea usar algoritmos genéticos donde la representación de la solución es una tira de bits donde hay  $(N+1)$  secuencias bits. Para cada tour-operador y el hotel tenemos asociadas una secuencia de bits que representa el número de habitaciones asignadas. Cada secuencia de bits ha de ser suficiente para codificar en binario el número total de habitaciones del hotel. Como solución inicial se asigna el mínimo de habitaciones a cada tour-operador y el resto al hotel. Los operadores a usar son los habituales de cruce y mutación. La función de evaluación valdrá infinito cuando no se cumplan todos los mínimos, y si se cumplen tendrá la siguiente fórmula:

$$h(n) = \text{num\_habitaciones}_H \frac{B_H}{Q_H} + \sum_{i=1}^N \text{num\_habitaciones}_i \frac{B_i}{Q_i}$$

3. Una empresa de transporte marítimo desea decidir la configuración de la carga de su próximo barco. La empresa recibe un conjunto de peticiones de envío de entre las que escoger. Cada petición tiene asociado un precio de transporte.

Cada petición va dentro de un tipo contenedor que está fijado por el tamaño, el peso del envío y las características de la carga. Existen solamente  $K$  tipos de contenedor. Los contenedores son propiedad de la empresa y su uso tiene un coste asociado que depende del tipo de contenedor.

Existen diferentes restricciones para la carga: la suma total de pesos de los contenedores no ha de sobrepasar la capacidad de carga del barco  $P_{max}$ , ni ha de ser inferior a cierto valor  $P_{min}$ . Cada contenedor tiene un peso asociado que depende de la carga que contiene. Las posibilidades de colocar los contenedores en el barco también imponen que haya un mínimo de  $C_{min}$  contenedores y un máximo de  $C_{max}$  contenedores de cada tipo.

El objetivo es encontrar la combinación de peticiones que maximicen el precio de transporte y minimicen el coste y que esté dentro de las restricciones impuestas.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone respecto a si es correcta, es eficiente, o es mejor o peor respecto a otras alternativas posibles. Justifica tu respuesta.

- (a) Se plantea solucionarlo mediante Hill-climbing utilizando como solución inicial el barco vacío y como operadores añadir y quitar contenedores del barco. Queremos usar como función de evaluación de las soluciones lo siguiente:

$$h(n) = \sum_{i=0}^{N_{cont}} Precio_i \times \sum_{i=0}^{N_{cont}} Coste_i$$

Donde  $N_{cont}$  es el número de contenedores que hay en una solución,  $Precio_i$  es el precio de transporte del contenedor  $i$  de la solución y  $Coste_i$  es el coste de un contenedor  $i$  de la solución.

- (b) Se plantea solucionarlo mediante Hill-climbing utilizando como solución inicial escoger al azar  $C_{min}$  contenedores de cada tipo. Como operador utilizamos intercambiar un contenedor del barco por otro que no esté en él. Queremos usar como función de evaluación de las soluciones lo siguiente:

$$h(n) = \sum_{i=0}^{N_{cont}} \frac{Precio_i}{Coste_i}$$

- (c) Se plantea utilizar algoritmos genéticos donde la representación de la solución es una tira de bits con tantos bits como peticiones haya, donde cada bit significa si la petición está o no en el barco. Para generar la población inicial usamos la misma técnica que en el apartado anterior. Usamos los operadores habituales de cruce y mutación y como función de evaluación usamos la del apartado anterior pero haciendo que valga cero si la solución incumple las restricciones del problema.

4. El mago Rincewind ha recibido un encargo de organizar una red de distribución de energía mágica en la región de Ankh-Morpork, para asegurar la cobertura de diversos artilugios mágicos. Por motivos de simplicidad, la región queda partida en un área de  $N \times M$  casillas, y en cada casilla se puede construir una torre (o no). Dependiendo del tipo de torre, esta ofrece más o menos área de cobertura mágica de forma circular con radio máximo  $r_{max}^t$ : de nuevo, por simplicidad, consideramos que una casilla está cubierta o no si la distancia euclídea desde el centro de la casilla al centro de la casilla con la torre es menor que  $r_{max}^t$ . Para cada casilla de esta rejilla, sabemos su importancia (por ejemplo, si es zona urbana es una casilla más prioritaria) como un valor  $v_{ij}$ . Hay tres tipos de torre: de retransmisión, media distancia y larga distancia. Cada una es más cara de construir que la anterior. Las torres de media y larga distancia ofrecen cobertura (p.ej.  $r_{max}^m = 4$  y  $r_{max}^l = 6$ , respectivamente), pero las de retransmisión no ( $r_{max}^{tr} = 0$ ). Todas las torres deben formar una red conexa, dos torres del tipo que sean se consideran conectadas si están a menos de distancia  $d_{max}$  (euclídea y contada desde el centro de cada casilla). Por este motivo, las torres de retransmisión que no ofrecen cobertura pueden ser útiles,

ya que son más baratas. Como problema final, el coste de construir una torre depende de la geografía de la región, así que cada casilla tiene un modificador de coste  $c_{ij}$  (entre 0.95 y 1.5). El coste final de construir una torre es el coste del tipo de torre multiplicado por el coste de la casilla.

El objetivo del problema es maximizar la calidad de la cobertura, contada esta como la suma de importancias de todas las casillas cubiertas, y minimizar el coste total de instalación de las torres.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone respecto a si es correcta, es eficiente, o es mejor o peor respecto a otras alternativas posibles para el mismo algoritmo. Justifica tu respuesta.

- (a) Se plantea solucionarlo mediante Hill-Climbing. Como solución inicial se parte de la solución vacía.

Como operadores se usan los siguientes: colocar\_torre que pone una torre de retransmisión en una casilla vacía concreta comprobando que (si no es la primera colocada) esté a menos distancia que  $d_{max}$  de otra torre, mejorar\_torre que transforma una torre de retransmisión en una de media distancia, o de media distancia en larga, y quitar\_torre que deja vacía la casilla donde estaba una torre. Como función heurística consideramos la suma de la calidad de todas las celdas con cobertura menos el coste multiplicado por un ponderador:

$$\sum_{i=1}^N \sum_{j=1}^N v_{ij} * \text{cubierta}_{ij} - \lambda * \sum_T \text{coste}(\text{tipo}(T)) * c_{\text{casilla}(T)}$$

- (b) Se plantea resolverlo mediante algoritmos genéticos. Para representar el problema utilizamos una tira de  $N * M * 3$  bits, donde cada grupo de 3 bits representa una casilla y su valor implica si está vacía (000) o si construiremos una torre de retransmisión (100), de media (010) o de larga distancia (001) en esa casilla. Como operadores usamos las tradiciones de cruce y mutación. El heurístico es el coste total de construcción menos la calidad total:

$$\sum_T \text{coste}(\text{tipo}(T)) * c_{\text{casilla}(T)} - \sum_{i=1}^N \sum_{j=1}^N v_{ij} * \text{cubierta}_{ij}$$

5. Se ha de celebrar un concurso nacional de canto coral que reune a varias corales no profesionales. Con el objetivo de reducir el desembolso que han de realizar los miembros de las corales, la organización ha solicitado y ha obtenido el patrocinio de una empresa para sufragar los costes de alojamiento de los participantes. Para llevar mejor el control del gasto en alojamiento, los organizadores del evento han de planificar las reservas de alojamiento de los 950 asistentes en los hostales de la zona. Disponen de una lista de hostales y para cada uno de ellos saben:

- A qué distancia se encuentra del lugar de la celebración del concurso,
- Cuantas camas tiene disponibles.
- El precio de la cama por noche.

Su problema es encontrar una asignación de las 950 personas a hostales de modo que no se supere el tope de camas disponibles en cada hostal, se minimice la suma de las distancias recorridas por todos los asistentes y se minimice la cantidad de dinero que la organización va a dedicar al pago del alojamiento. Se puede suponer que sólo hay que planificar una noche, que en cada hostal todas las camas cuestan lo mismo y que solo hemos de asignar personas a hostales (la asignación de personas a camas dentro del hostal la gestionará el propio hostal con los clientes por orden de llegada).

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 (estado o solución inicial, operadores, función heurística,...). Para cada una de ellas se ha de comentar los diferentes elementos de la propuesta, analizando si la técnica escogida es adecuada para este problema, si cada uno sus elementos son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

- (a) Usar Hill climbing. Como solución inicial se utiliza una estrategia avariciosa poniendo a las personas en los hostales con el cociente precio/distancia más bajo. Como operador se utiliza mover un grupo de personas de un hostal a otro, moviendo tantas como quepan en el hostal destino. La función heurística es la suma, para todas las personas, del producto entre la distancia recorrida por una persona y el precio de su cama.
- (b) Usar algoritmos genéticos. Representaremos la solución como una tira de bits donde asignamos a cada hostal tantos bits como se necesiten para representar el número máximo de personas que caben. Para generar la población inicial se genera cada individuo distribuyendo aleatoriamente todas las personas entre los hostales. Utilizamos los operadores habituales de cruce y mutación. La función heurística es la suma de las distancias de todos los hostales que tienen alguna persona asignada, dividida por la suma de los precios de las camas de todas las personas.
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  $B$ .
- (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.
- (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 correspondientes a los  $B$  aparatos.
- (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.
- (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.
7. 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  $r_i$  sabemos su localización  $l_i$  en el campus (supondremos que tenemos un sistema de coordenadas en el que podemos situar cada router) y el ancho de banda

$bw_i$  del tráfico directo que debe distribuir (el que no le llega de otros routers). También conocemos las coordenadas del equipo de comunicaciones ( $l_{EC}$ ). 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 ( $bw_A < bw_B < bw_C$ ). 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  $r_i$  necesitamos instalar un cable  $c_i$  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. 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 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,...). Comenta muy brevemente la solución que se propone respecto a si es correcta, eficiente o mejor/peor respecto a otras alternativas posibles. Justifica tus 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 el sumatorio, para todos los routers, del coste del router más el coste del cable que lo conecta (a otro router o al equipo de comunicaciones).
  - (b) 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 operadores 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 de un router. La función heurística quiere reducir al máximo la longitud de las conexiones y por ello es el sumatorio, para todos los routers, del coste del router más el coste del cable que lo conecta a otro router o al equipo de comunicaciones dividido por la longitud de esa conexión.
  - (c) Usar 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 (es importante tener en cuenta que  $r+1$  puede ser menor que  $2^n$ ). Como operadores usamos los operadores de cruce y mutación habituales. Para generar la población inicial usamos el método del apartado b). La función heurística es la suma ponderada del coste de cada router (con peso 0,7) y el coste de su cable de conexión (con peso 0,3) para darle mayor relevancia al coste del router (que es lo más caro).
8. La International Telecommunications Union (ITU) es un agencia de Naciones Unidas que regula el uso de las tecnologías de la telecomunicación y entre otras cosas coopera en la asignación de órbitas para satélites. El incremento del uso de satélites de todo tipo ha llevado a buscar métodos automáticos para asignar posiciones en órbitas que mantengan la seguridad de los satélites y la calidad de su funcionamiento.

Para solucionar el problema, se ha dividido la órbita geosincrónica en regiones donde se asignarán las posiciones orbitales de los satélites. Obviamente, una región del espacio tiene un volumen. Se quiere resolver el problema para cada región.

Hay  $S$  diversos tipos de satélites especializados que se desean poner en órbita (televisión, telefonía, científicos, militares, ...). Para cada satélite se tienen dos informaciones, el espacio de seguridad (volumen necesario para que el satélite pueda navegar con el mínimo riesgo de colisión y pueda alimentar sus paneles solares) y el coste de alquiler de la órbita que cobra la ITU al dueño del satélite (este precio depende de cada satélite y no es una función del espacio de seguridad).

Las restricciones que tiene la ITU son que la suma del volumen que han de ocupar los espacios de seguridad de los satélites ha de cubrir como mínimo 1/3 del volumen de la región si se quieren cubrir

las necesidades existentes, pero no puede superar los  $2/3$  si no se quieren tener problemas con otros lanzamientos que deben atravesar la órbita geosincrónica o con la basura espacial. También se ha de cubrir un cupo mínimo  $m_i$  para cada tipo de satélites. El objetivo del problema es maximizar la ganancia que obtiene la ITU con el alquiler de órbitas.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone respecto a si es correcta, es eficiente, o es mejor o peor respecto a otras alternativas posibles. Justifica tu respuesta.

- (a) Se plantea usar Hill-climbing, como solución inicial consideramos que asignamos aleatoriamente al menos la mínima cantidad de satélites para cubrir la cuota de cada tipo hasta ocupar  $1/3$  del volumen de la región. Disponemos del operador `añadir-satélite()` que comprueba el posible exceso de ocupación en la región y del operador `quitar-satélite()` que comprueba la posible infra-ocupación de la región. Como heurística usamos la suma para todos los satélites en la solución del producto entre el espacio de seguridad del satélite y el coste de alquiler del satélite.
  - (b) Se plantea usar Hill-climbing, como solución inicial ordenamos los satélites descendente por el coste del alquiler y vamos añadiendo los satélites en ese orden hasta llenar  $2/3$  del volumen. Como operador utilizamos cambiar un satélite de un tipo por otro del mismo tipo que no esté ya en la solución. Como función heurística usamos la suma del espacio de seguridad de todos los satélites en la solución multiplicado por la suma del coste del alquiler de todos los satélites.
  - (c) Se plantea usar algoritmos genéticos. Un individuo es una tira de bits cuya longitud es el número total de satélites que tenemos. Si el bit está a 1 significa que el satélite está en la solución y si está a 0 es que no lo está. La población inicial se genera creando individuos que tengan a 1 los bits de los  $m_i$  satélites con mayor coste de alquiler de cada tipo. Como operadores se usan los habituales de cruce y mutación. Como función heurística usamos la suma del coste de alquiler de todos los satélites en la solución.
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 kilómetros 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  $i$ .

- (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:

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

- (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:

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

- (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.
10. Los dueños del hotel “MiniPreu” quieren reducir su presupuesto de limpieza para maximizar sus ganancias. Para hacerlo han decidido tener tres tipos diferentes de precios para sus habitaciones. El precio A incluye una limpieza diaria de la habitación y tiene un precio de  $pa$  euros por día, el precio B incluye una limpieza de la habitación en días alternos y tiene un precio de  $pb$  euros por día y el precio C solo incluye una limpieza el primer y último día y tiene un precio de  $pc$  por día. El mínimo número de días que se pueden reservar para una habitación es de dos, y asumiremos que la limpieza de las habitaciones de precio B esta organizada de manera que la última limpieza se hace el día en el que el cliente deja su habitación. Tenemos un total de  $R$  habitaciones en el hotel.

Limpiar una habitación de precio A cuesta  $ca$  euros por día de limpieza, una habitación de precio B cuesta  $cb$  por día de limpieza y una habitación de precio C cuesta  $ca$  más el número de días que la habitación ha sido reservada multiplicado por el coste  $cc$ . Las habitaciones que están vacías no se limpian.

El departamento de sanidad impone como restricción que toda habitación (ocupada o no) ha de limpiarse al menos 10 veces al mes. Esto fuerza a que ninguna habitación pueda quedar sin ocupar muchos días.

El hotel recibe peticiones de reserva para un mes completo, cada reserva incluye el precio que el cliente quiere, el día en el que la reserva comienza y el número de días a reservar. Asumiremos que todas las peticiones comienzan y finalizan dentro del mes. El objetivo es decidir qué reservas aceptar, de manera que los gastos de limpieza se minimicen y las ganancias se maximicen.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

- (a) Queremos usar Hill-climbing, la solución inicial es la solución vacía. Como operadores de búsqueda usamos **añadir-reserva** que añade una reserva de una habitación, **cambiar-reserva** que cambia la reserva de una habitación por otra reserva que no esté ya en la solución. La función heurística es la suma de todos los costes de limpieza de las habitaciones multiplicado por las ganancias obtenidas por las reservas.

- (b) Queremos usar Hill-climbing, la solución inicial se obtiene de la siguiente manera: para una habitación elegimos una de las reservas que tenga la fecha más temprana, la siguiente reserva será la que tenga la fecha más temprana que empiece después del final de la última reserva asignada a la habitación, repetimos este procedimiento hasta que no podamos asignar más reservas a la habitación. Hacemos esto para todas las habitaciones. Como operadores de búsqueda usamos **cambiar-reserva** que cambia una reserva de una habitación por otra reserva que no esté en la solución. La función heurística es la suma de todos los costes de limpieza de las habitaciones menos las ganancias obtenidas por las reservas.
- (c) Se plantea utilizar algoritmos genéticos. Asignamos a cada reserva un número de 0 a  $R$ , la concatenación de estos números para todas las reservas en binario es la codificación de una solución. El número 0 significa que la reserva no está asignada a ninguna habitación, otro número representa el número de habitación asignada a la reserva. Para generar la población inicial obtenemos una solución asignando aleatoriamente una reserva a cada habitación (solo  $R$  reservas tienen un número diferente de 0). Como operadores genéticos usamos los operadores habituales de cruce y mutación. La función heurística es la suma para todas las reservas en la solución del cociente entre las ganancias obtenidas por la reserva y los costes de limpieza de la reserva.
11. Los alcaldes de las poblaciones del Baix Llobregat se han unido para conseguir el compromiso de la Generalitat de construir una nueva línea de metro que pueda servir a zonas que aún no cubre la red actual de metro y trenes. Para realizar el proyecto los ingenieros del Área Metropolitana de Barcelona han pre-seleccionado  $L$  lugares potenciales donde construir las estaciones de metro y han determinado, a partir del análisis de los datos de movilidad, el número de usuarios diarios que usaría cada estación potencial. Pero por limitaciones presupuestarias, la línea de metro solo puede tener  $E$  estaciones, a escoger entre los  $L$  lugares potenciales. El objetivo es determinar qué  $E$  lugares escoger para las estaciones y el orden en el que se han de conectar de manera que se sirva al mayor número de personas y el recorrido total de la línea sea el menor posible.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone respecto a si es correcta, es eficiente, o es mejor o peor respecto a otras alternativas posibles. Justifica tu respuesta.

- (a) Se plantea solucionarlo mediante Hill-climbing. Como solución inicial elegimos una solución sin estaciones. Como operadores de búsqueda usamos **añadir-estación**, que añade una estación al final del recorrido de la solución e **intercambiar-estaciones** que intercambia dos estaciones dentro del recorrido. La función heurística es:

$$h(n) = \sum_{i=1}^{E-1} dist(E_i, E_{i+1}) + \sum_{i=1}^E Personas(E_i)$$

Donde  $dist(E_i, E_{i+1})$  es una función que indica la distancia (en metros) entre la estación  $E_i$  y la siguiente, y  $Personas(E_i)$  es una función que devuelve el número de usuarios diarios que usaría la estación  $E_i$ .

- (b) Usar Hill climbing. Como solución inicial escogemos una estación al azar, usamos esta estación como referencia y escogemos como siguiente estación la más cercana a esta, pasamos a tomar como referencia esta nueva estación y buscamos la más cercana que no esté en la solución, repetimos lo mismo hasta tener  $E$  estaciones. Como operadores de búsqueda usamos el **cambiar-estación** que intercambia una estación de la solución por otra que no esté en la solución e **intercambiar-estaciones** que intercambia dos estaciones dentro del recorrido. La función heurística es:

$$h(n) = \frac{\sum_{i=1}^{E-1} dist(E_i, E_{i+1})}{\sum_{i=1}^E Personas(E_i)}$$

- (c) Se plantea utilizar algoritmos genéticos. Asociamos a cada lugar posible con un número del 1 al  $L$ . La codificación de la solución se realiza considerando que tenemos una tira de bits de longitud

$E \times \log_2(L)$ , donde concatenamos los identificadores de los lugares que hay en la solución, el orden en el que aparecen en la codificación es el orden del recorrido. Para generar la población inicial se escogen  $E$  lugares al azar de los  $L$  y se colocan al azar en la solución. Como operadores genéticos usamos los operadores habituales de cruce y mutación. La función heurística es:

$$h(n) = \alpha \times \sum_{i=1}^{E-1} dist(E_i, E_{i+1}) + (1 - \alpha) \times \left( -\sum_{i=1}^E Personas(E_i) \right)$$

Donde  $\alpha$  es un valor entre 0 y 1 que permite dar más o menos importancia a cada criterio.

12. Una empresa de distribución de electricidad necesita calcular cada día qué centrales de producción ha de tener en marcha para abastecer la demanda de sus clientes. La empresa dispone de  $C$  centrales de producción eléctrica ubicadas en diferentes puntos de la geografía, con una capacidad de producción  $CA_i$  cada una. El coste diario de tener en marcha cada central es  $CO_i$ .

El contrato que tiene la empresa con cada uno de sus  $CL$  clientes indica la cantidad de electricidad ( $E_i$ ) que se les debe suministrar cada día. La distancia entre los clientes y las centrales ( $d_{ij}$ ) supone una pérdida en la eficiencia del suministro, esta pérdida se calcula como un factor fijo  $P$  multiplicado por esa distancia.

El objetivo es determinar el conjunto de centrales que se han de tener en marcha y a qué clientes han de servir, de manera que se minimice el coste de tenerlas en marcha y la pérdida debida a la distancia, teniendo en cuenta que la cantidad de electricidad que reciben los clientes de una central no ha de sobrepasar su capacidad de producción y que se ha de servir toda la demanda de los clientes. Asumiremos que los valores asignados al problema siempre permiten hallar una solución.

Una posible solución a este 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) Se plantea solucionarlo mediante Hill-climbing utilizando como solución inicial asignar a cada cliente al azar una central. Como operadores se utiliza el mover un cliente de una central a otra e intercambiar dos clientes entre dos centrales. La función heurística es la siguiente:

$$h(n) = \sum_{i=1}^C (CO_i \times marcha(i)) - \sum_{i=1}^C \sum_{j=1}^{CL} asig(i,j) \times d_{ij}$$

donde  $asig(i,j)$  es una función que retorna uno si el cliente  $i$  está asignado a la central  $j$  y cero en otro caso y  $marcha(i)$  retorna uno si la central tiene clientes asignados y cero si no.

- (b) Se plantea solucionarlo mediante Hill-climbing utilizando como solución inicial un algoritmo avaricioso que ordena ascendenteamente las centrales según su coste y los clientes según su demanda, y va asignando clientes a las centrales según ese orden. Cuando se sobrepasa la capacidad de una central se pasa a la siguiente. El operador utilizado es intercambiar dos clientes entre dos centrales siempre que no se supere la capacidad de suministro de alguna de ellas. La función heurística es:

$$h(n) = \left( \sum_{i=1}^C CO_i \right) \times \left( \sum_{i=1}^C \sum_{j=1}^{CL} P \times asig(i,j) \times d_{i,j} \right)$$

- (c) Se plantea utilizar algoritmos genéticos donde la representación de la solución es una tira de bits. Esta está compuesta por la concatenación de  $C$  grupos de  $CL$  bits, donde cada grupo representa la asignación o no de cada cliente a la central como un 1 o un 0. Las poblaciones iniciales las obtenemos

mediante el mismo mecanismo que en el apartado anterior. Se utilizan los operadores habituales de cruce y mutación. La función heurística es:

$$h(n) = \left( \sum_{i=1}^C CO_i \times marcha(i) \right) + \left( P \times \sum_{i=1}^C \sum_{j=1}^{CL} asig(i,j) \times d_{i,j} \right)$$

13. Despues de los incendios ocurridos en el verano se nos plantea el problema de buscar una forma efectiva de repoblar las áreas afectadas. Dada un área concreta de cierto número de hectáreas a repoblar, dividimos el área en una cuadrícula de  $N \times M$  áreas y nos planteamos cuanto plantar en cada área.

Cada área  $i$  de la cuadrícula tiene asociado un factor de repoblación ( $Fr(i)$ ) que va de 0 a 3 (0 significa ninguna repoblación, 1 significa plantar  $A$  árboles, 3 significa plantar  $3 \times A$  árboles). Disponemos un máximo de  $K \times A$  árboles para plantar, pero no queremos plantar menos de  $I \times A$  árboles.

Además queremos minimizar el riesgo de incendio del área. Este es una función del factor de repoblación de un elemento de la cuadrícula y todos los elementos adyacentes, y se calcula de la siguiente manera:

$$R(i) = \sum_{\forall j \text{ adyacente a } i} Fr(i) \times Fr(j)$$

El objetivo es encontrar una solución que plante el máximo número de árboles (entre los límites  $I \times A$  y  $K \times A$ ) y con el mínimo riesgo de incendio.

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. Para ello generamos una solución inicial en la que todas las áreas tienen factor de repoblación 0. Los operadores son aumentar o disminuir el factor de repoblación de un área. La función heurística que queremos optimizar es:

$$h'(n) = \sum_{i=1}^{M \times N} R(i) - \sum_{i=1}^{M \times N} Fr(i)$$

- (b) Usar Hill Climbing. Para ello generamos una solución inicial en la que asignamos secuencialmente factor de repoblación +1 a cada elemento de la cuadrícula hasta llegar a la cantidad mínima de árboles que hemos de plantar. Los operadores son aumentar el factor de repoblación siempre que este no sea 3 o disminuir el factor de repoblación de un área siempre que no sea 0. La función heurística que queremos optimizar es:

$$h'(n) = \left( \sum_{i=1}^{M \times N} Fr(i) - K \right) - \frac{1}{\sum_{i=1}^{M \times N} R(i)}$$

- (c) Usar algoritmos genéticos. Para representar una solución generamos una tira de  $2 \times M \times N$  bits (con 2 bits codificamos el factor de repoblación). Como solución inicial utilizamos la misma que en el apartado anterior. Como operadores utilizamos los operadores habituales de cruce y mutación. La función heurística que queremos optimizar es:

$$h'(n) = \sum_{i=1}^{M \times N} R(i) \times \sum_{i=1}^{M \times N} Fr(i)$$

14. Una de las labores de los equipos de comunicaciones en una red es distribuir el flujo de paquetes que le llegan a un equipo entre los diferentes equipos con los que está conectado, optimizando la calidad de la distribución de paquetes de forma local.

Una posible estrategia de distribución de paquetes sería decidir a priori cuantos de los paquetes que se reciben en cierto segundo se envían por cada conexión, sin preocuparnos exactamente de a donde deben ir.

Un equipo de comunicaciones recibe cierto número de paquetes por segundo ( $P$ ) que tiene que distribuir entre sus conexiones de salida. Para cada una de las  $N$  conexiones de salida del equipo se conocen tres informaciones: su capacidad (en número de paquetes por segundo), el tiempo de retraso que introduce la conexión (tiempo medio adicional que añade el nodo de salida al tiempo de llegada a destino de cada paquete) y el número medio de saltos que hará cada paquete hasta llegar a su destino.

Deseamos calcular el número de paquetes que debemos enviar por cada conexión de salida para optimizar la distribución de paquetes de manera que el retraso y número medio de saltos de los paquetes sea el mínimo posible. Evidentemente se han de enviar todos los paquetes que llegan.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone respecto a si es correcta, es eficiente, o es mejor o peor respecto a otras alternativas posibles. Justifica tu respuesta.

- (a) Se plantea solucionarlo mediante Hill-Climbing usando como solución inicial el repartir todos los paquetes a partes iguales entre todas las conexiones de salida del equipo. Como operador utilizamos mover cierta cantidad de paquetes de una conexión a otra. Como función heurística usamos el sumatorio, para todas las conexiones con paquetes asignados, del producto entre el retraso de la conexión y el número de saltos.
  - (b) Se plantea solucionarlo mediante Hill-Climbing tomando como solución inicial el repartir los paquetes entre las conexiones de salida del equipo asignando aleatoriamente a una conexión el máximo de su capacidad, repitiendo el proceso hasta haber repartido todos los paquetes. Como operador se utilizaría mover cualquiera de los paquetes de una conexión a otra siempre que la operación sea válida. Como función heurística usamos el sumatorio para todas las conexiones con paquetes asignados del producto entre el número de paquetes asignados a la conexión y el retraso de la conexión más el sumatorio, para todas las conexiones con paquetes asignados, del producto entre el número de paquetes asignados a la conexión y el número de saltos medio de esa conexión.
  - (c) Se plantea solucionarlo mediante Algoritmos Genéticos, donde la representación del problema es una tira de bits compuesta por la concatenación de la representación en binario del número de paquetes asignados a cada conexión (evidentemente usando el mismo número de bits para cada conexión). Como solución inicial ordenamos las conexiones por su retraso (de menor a mayor) y asignamos paquetes en ese orden llenando la capacidad de cada conexión hasta haber asignado todos los paquetes. Usamos los operadores de cruce y mutación habituales. Usamos como función heurística el sumatorio, para todas las conexiones, del producto entre el número de paquetes asignados a cada conexión y el retraso de la conexión.
15. La Agència de Gestió d'Ajuts Universitaris (AGAUR) de la Generalitat de Catalunya quiere organizar mejor la asignación de ayudas destinadas a proyectos de investigación orientados a mejorar la gestión sanitaria del COVID-19. Para ello AGAUR planea realizar  $C$  convocatorias en los próximos meses. Cada convocatoria de ayudas tiene un presupuesto a repartir entre las peticiones concedidas. A estas convocatorias los grupos de investigación pueden presentar peticiones (la misma petición puede presentarse a hasta tres convocatorias distintas, pero solo se puede conceder una vez). Una petición tiene un presupuesto solicitado y AGAUR le asigna una prioridad entre 1 y 3 dependiendo del historial del grupo investigador. Hay un total de  $P$  peticiones que han pasado ya un filtro previo realizado por un grupo de expertos, asegurando que todos los proyectos de investigación son relevantes y de calidad.

AGAUR quiere asignar la mayor cantidad de dinero posible en cada convocatoria, pero haciendo que globalmente la proporción de grupos con concesiones de ayuda cumpla las siguientes restricciones: como máximo el 30% de concesiones ha de ser para los grupos de prioridad 2 y como máximo el 10% para los de prioridad 3. Obviamente la suma de los presupuestos solicitados de las peticiones concedidas en una convocatoria no puede superar el presupuesto de esa convocatoria.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

- (a) Usar Hill-climbing. Para hallar la solución inicial recorremos todas las convocatorias y asignamos tantas peticiones de prioridad 1 solicitadas a esa convocatoria como podamos, sin exceder su presupuesto. Como operadores utilizamos: *asignar* una petición a una convocatoria, respetando las restricciones de máximos de peticiones concedidas por prioridad; *move* una petición asignada en una convocatoria  $C_i$  a otra convocatoria  $C_j$ , comprobando que no se supera el presupuesto máximo de  $C_j$ ; e *intercambiar* una petición  $P_x$  asignada a una convocatoria por otra petición  $P_y$  que se haya presentado a esa convocatoria pero que no haya sido asignada aún, comprobando que no se supera el presupuesto máximo de la convocatoria. Como función heurística usamos:

$$h1(n) = \frac{\sum_{\forall conv_j} \sum_{\forall pet_i \in conv_j} Presupuesto(pet_i)}{\sum_{\forall conv_j} Presupuesto(conv_j)}$$

donde  $pet_i \in conv_j$  significa que la petición i ha sido concedida (asignada) en la convocatoria j.

- (b) Usar algoritmos genéticos. Codificamos el problema con  $C \times P$  bits donde cada grupo de  $P$  bits corresponde a una convocatoria de manera que un bit a '1' significa que la petición ha sido concedida en esa convocatoria. Las soluciones iniciales las generamos utilizando el método del apartado anterior. Como operadores genéticos usamos los operadores habituales de cruce y mutación. Como función heurística usamos:

$$h2(n) = \sum_{\forall conv_j} \left( Presupuesto(conv_j) - \sum_{\forall pet_i \in conv_j} Presupuesto(pet_i) \right)$$

donde  $pet_i \in conv_j$  significa que la petición i ha sido concedida (asignada) en la convocatoria j.

16. 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 kilómetro. El punto inicial del recorrido diario es el garaje de furgonetas con el tanque lleno de hidrógeno y el punto final es el mismo garaje de furgonetas (donde rellenan el tanque y revisan la mini-furgoneta para que esté perfecta el día siguiente). Los puntos intermedios del recorrido solo pueden ser parques o surtidores.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (estado o solución inicial, operadores, función heurística,...). Para cada una de ellas se ha de comentar los diferentes elementos de la propuesta, analizando si la técnica escogida es adecuada para este problema, si cada uno sus elementos son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

- (a) Queremos utilizar A\* para minimizar el número de kilómetros recorridos por la furgoneta y reducir el número de repostajes al mínimo. Partimos de un estado inicial vacío. El operador propuesto es pasar del punto actual a otro punto. Utilizaremos como función de coste los litros de hidrógeno consumidos (se calcula a partir de los kilómetros recorridos del punto actual al escogido, excepto cuando el punto escogido es un surtidor, ya que en ese caso el coste son los litros de hidrógeno consumidos del punto actual al surtidor escogido menos los litros de hidrógeno cargados en el repostaje). 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 estima los litros de hidrógeno por consumir calculando la suma de las distancias de los puntos por recorrer al punto actual y multiplicando el resultado por los  $x$  litros de hidrógeno por kilómetro que consume.
- (b) Queremos utilizar Hill Climbing para minimizar el número de kilómetros recorridos por la mini-furgoneta y reducir el número de repostajes al mínimo. Partimos de una solución inicial en la que se genera aleatoriamente un orden de visita de los parques y se intercala un paso por un surtidor de hidrógeno al azar entre parque y parque. Se dispone de dos operadores de modificación de la solución: uno para eliminar del camino un paso por un surtidor, y otro para modificar el orden en el que visitamos alguno de los puntos del camino (ya sea parque o surtidor), desplazándolo adelante o atrás en el orden una cantidad  $n$  de posiciones. La función heurística es la longitud del camino recorrido + 2 kilómetros extra de penalización por cada paso por un surtidor.
17. La empresa *Cloud For U* se dedica a revender tiempo de cómputo para cloud computing y para ello ha montado un servicio que recibe solicitudes para usar ese tiempo. Este servicio es un sistema de pre-reserva que nos permite saber las solicitudes para un mes completo. Cada petición nos indica un día y hora de inicio y un tiempo de uso en horas, indicando además la prioridad de la petición (de 1 a 3, siendo 1 la menos prioritaria). Una petición ha de ejecutarse completamente, sin interrupciones y usando exclusivamente una CPU. *Cloud For U* compra el tiempo de cómputo en una CPU durante un mes a  $M$  euros y cobra el tiempo de cómputo a  $prioridad \times P$  euros/hora, con el compromiso de que si no se sirve la petición, ésta se servirá seguro el mes siguiente a solo  $P$  euros/hora independientemente de su prioridad inicial.

La empresa nos plantea saber cuantas CPUs debe alquilar en el mes actual para poder obtener el mayor beneficio posible, intentando reducir el número de peticiones diferidas al mes siguiente.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística,...). Para cada una de ellas se ha de comentar la solución que se propone, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

- (a) Se plantea aplicar Hill-climbing, para crear la solución inicial suponemos que necesitamos tantas CPUs como peticiones nos han llegado, de manera que asignamos cada petición a una CPU diferente. Como operador usamos fusionar dos CPUs, que consiste en combinar las peticiones de dos CPUs en una sola eliminando la CPU sobrante. Como función heurística consideraremos que alquilaremos las CPUs en la solución que tengan más del 90% de ocupación ( $CPU90$ ), por lo tanto minimizaremos la función:

$$h(n) = (N_{CPU} \times M) - \sum_{\forall p_i \in CPU90} prio(p_i) \times P \times long(p_i) + \lambda \times \sum_{\forall p_i \in \overline{CPU90}} P \times long(p_i)$$

donde  $N_{CPU}$  es el número de CPUs usadas en la solución,  $CPU90$  es el conjunto de CPUs ocupadas a más del 90% y  $\overline{CPU90}$  es su conjunto complementario,  $long(p_i)$  es la longitud en horas de la petición y  $prio(p_i)$  es la prioridad de la petición.  $\lambda$  es un peso para ajustar la penalización de las tareas que se difieren al mes siguiente.

- (b) Se plantea usar algoritmos genéticos, como representación escogemos una cadena de bits donde para cada petición se tiene un número en binario que es la CPU que tiene asignada; suponemos que ese número es un valor dentro del rango ( $NP_3 \dots NC + 1$ ), donde  $NP_3$  es el número de peticiones de prioridad 3 y  $NC$  es el numero total de CPU's, añadimos un valor adicional para representar una CPU ficticia (a la que se asignan las peticiones diferidas al mes siguiente). Para generar una solución se van generando al azar valores entre los límites mencionados y se asignan consecutivamente a las peticiones de la siguiente manera, con un 50% de probabilidad se le asigna el valor de una CPU real y con un 50% de probabilidad se le asigna la CPU ficticia. Utilizamos los operadores habituales de cruce y mutación. Como función heurística consideraremos que alquilaremos las CPUs en la solución que no sean la ficticia, por lo tanto minimizaremos la función:

$$h(n) = (N_{CPU} \times M) - \sum_{\forall p_i \notin CPU_f} prio(p_i) \times P \times long(p_i) + \sum_{\forall p_i \in CPU_f} P \times long(p_i)$$

donde  $N_{CPU}$  es el número de CPUs que no son la ficticia y que tienen peticiones asignadas,  $CPU_f$  es el conjunto de peticiones que están en la CPU ficticia,  $long(p_i)$  es la longitud en horas de la petición y  $prio(p_i)$  es la prioridad de la petición.

- (c) Usar el algoritmo de A\*. Definimos el estado como la asignación de peticiones a CPU's. Partimos de la asignación vacía en la que hay cero CPU's y cero peticiones asignadas. Tendremos un operador para añadir una CPU (comprueba que no estuviera ya en la solución), el coste del operador es  $M$ . Tendremos otro operador para añadir una petición a una CPU (comprueba que la petición no haya sido añadida a la solución, y que la CPU ya ha sido añadida previamente con el otro operador), su coste es  $-prio(p_i) \times P \times long(p_i)$ , donde  $long(p_i)$  es la longitud en horas de la petición y  $prio(p_i)$  es la prioridad de la petición  $p_i$  que colocamos. La función heurística es la suma de las longitudes en horas de las peticiones que nos quedan por colocar en CPU's, expresada así:

$$h(n) = \sum_{\forall p_i \notin P_{assig}} long(p_i)$$

donde  $P_{assig}$  es el conjunto de peticiones ya asignadas.

18. Como una de sus medidas de reactivación económica para salir de la crisis provocada por la COVID-19, y aprovechando la gran proporción de población jubilada que ya está vacunada, el Ministerio de Sanidad, Consumo y Bienestar Social quiere proporcionar viajes gratuitos a los jubilados a través del IMSERSO. Para ello, el Ministerio intenta repartir a los  $J$  jubilados que se apuntan a la iniciativa anualmente entre los  $H$  hoteles que se han presentado al plan de ayudas del sector por la COVID-19 (no hay restricciones sobre  $J$  y  $H$ , pero claramente  $J \gg H$ ). Cada jubilado  $j$  puede tener o no una discapacidad de tipo K(cognitiva), M(medicalización continua) o R(movilidad reducida), y puede presentar al IMSERSO un máximo de tres lugares a los que prefiere viajar. Por otra parte, cada hotel  $h$  está asociado a un lugar, dispone de un número de plazas  $PL_h$  a un precio único (cada hotel tiene un precio por plaza  $PR_h$ ) y puede estar habilitado para discapacidades de tipo K, M y/o R, o no estarlo para ninguna.

El reparto debe:

- maximizar el número de jubilados con plaza asignada,
- minimizar el precio total que se deberá gastar el Ministerio,
- priorizar la asignación de jubilados que han viajado menos a través del IMSERSO.
- intentar respetar al máximo las preferencias de los jubilados,
- respetar las discapacidades de los jubilados.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

Sea:

- $A = \#$ jubilados asignados a plazas,
- $B = \#$ jubilados con preferencias satisfechas,
- $C =$ suma de frecuencias de viajes de los jubilados asignados,
- $D =$ precio total de la asignación,
- $E =$ suma de los precios de todas las plazas de todos los hoteles,
- $F =$ número total de plazas de los H hoteles.

- (a) Queremos usar Hill-climbing, tomando como solución inicial la asignación de  $F$  jubilados con menos frecuencia de viajes, intentando llenar primero los hoteles más baratos. Como operador de búsqueda usamos `intercambiar(plaza, jubilado1, jubilado2)` donde *jubilado1* está asignado a la plaza en la solución actual, *jubilado2* no está asignado a la solución actual, y la *plaza* cumple las posibles discapacidades del *jubilado2*. La función heurística será:

$$h'(n) = A + B + C + D$$

- (b) Se plantea utilizar algoritmos genéticos. Asignamos a cada plaza de hotel un número de 0 a  $F$ , y representamos una solución como una secuencia de  $J \cdot \log_2(F+1)$  bits, representando para cada jubilado  $j$  la plaza de hotel asignada, o el valor 0 si no tiene plaza asignada. Para generar la población inicial obtenemos una solución asignando aleatoriamente un jubilado a cada plaza (solo  $F$  jubilados tienen un número diferente de 0). Como operadores genéticos usamos los operadores habituales de cruce y mutación. La función heurística es:

$$h'(n) = \frac{A + B}{\frac{C}{10*J} + \frac{D}{E}}$$

19. El domingo 28 de Mayo se celebrarán las elecciones autonómicas y municipales en Cataluña, tarea que conlleva de por sí un conjunto de dificultades logísticas que, tras cambios de protocolo con la rebaja de casos de Covid19 en 2023, requieren reseleccionar todos los centros de votación teniendo en cuenta las nuevas restricciones. Para ello nos proporcionan un mapa del territorio con  $L$  posibles locales donde podemos colocar un total de  $C$  colegios electorales ( $L > C$ ). Tenemos una tabla con las distancias entre los locales. Además, disponemos para cada local, el número de personas que podrían ser seleccionadas durante el sorteo de vocales y presidentes de mesa.

El objetivo es colocar todos los colegios electorales de manera que se maximice la suma de las distancias entre ellos (para obtener así la cobertura máxima del territorio) y se maximice también el número de personas disponibles que cada centro tiene para su sorteo de vocales y presidentes. Asumimos que cada centro tiene un número constante de personas para el sorteo (dado en una tabla), que ese número puede ser diferente para cada centro y que no cambia en función de qué otros centros se acaban escogiendo.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone respecto a si es correcta, es eficiente, o es mejor o peor respecto a otras alternativas posibles. Justifica tu respuesta.

- (a) Se plantea solucionarlo mediante Hill-Climbing. Como solución inicial se parte de la solución vacía. Como operadores se usa el colocar un colegio electoral en un local vacío y quitar un colegio electoral de un local. Como función heurística se usa la suma de las distancias de los locales de cada uno de los  $C$  colegios electorales al resto, multiplicado por la suma de las personas disponibles para el sorteo en cada local asignado.

- (b) Se plantea resolverlo mediante algoritmos genéticos. Para representar el problema utilizamos una tira de  $L$  bits, donde un bit a 0 indica que ese local no tiene colegio electoral asignado y un bit a 1 indica que si se le ha asignado un colegio electoral. Como población inicial generamos aleatoriamente  $n$  individuos donde en cada uno hay exactamente  $C$  bits a 1. La función heurística es la suma de distancias de cada colegio electoral al resto más una constante  $k$  por la suma de las personas disponibles para el sorteo en cada local asignado. Como operadores usamos un operador de cruce que intercambia aleatoriamente la mitad de los bits de cada individuo de la pareja cruzada y un operador de mutación que intercambia aleatoriamente dos posiciones de la tira de bits cumpliendo que en una sea un bit a 1 y en la otra un bit a 0.

Comenta cada una de las soluciones que se proponen, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

20. La guerra de Rusia contra Ucrania está creando una crisis humanitaria por los cientos de miles de desplazados que huyen del conflicto. Uno de los países con mayor dificultad para gestionar esta crisis es la República de Moldavia, un país con solo dos millones y medio de habitantes, tercer país más pobre de Europa y que tiene frontera con Ucrania y Rumanía. Aunque la mayoría de ucranianos huyen hacia Polonia, miles y miles de desplazados atraviesan la frontera moldava por nueve lugares diferentes. El gobierno moldavo nos pide ayuda para organizar a los voluntarios, los espacios de acogida y los limitados recursos que tienen, ante la falta de ayuda de las grandes organizaciones humanitarias que están centrando sus esfuerzos en la frontera polaca.

El gobierno moldavo nos proporciona un mapa del territorio con  $L$  posibles locales donde podemos colocar un total de  $C$  centros de acogida ( $L > C$ ). Tenemos también una tabla con las distancias entre los locales. Además también disponemos del número de voluntarios que viven cerca de cada local en el que se puede ubicar un centro de acogida (los voluntarios que potencialmente pueden venir a prestar su ayuda en ese local).

El objetivo del sistema es colocar todos los centros de acogida de manera que se maximice la suma de las distancias entre ellos (para obtener así la cobertura máxima del territorio moldavo) y se maximice también el número de voluntarios disponibles (voluntarios que viven cerca y que podrán venir a prestar su ayuda).

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

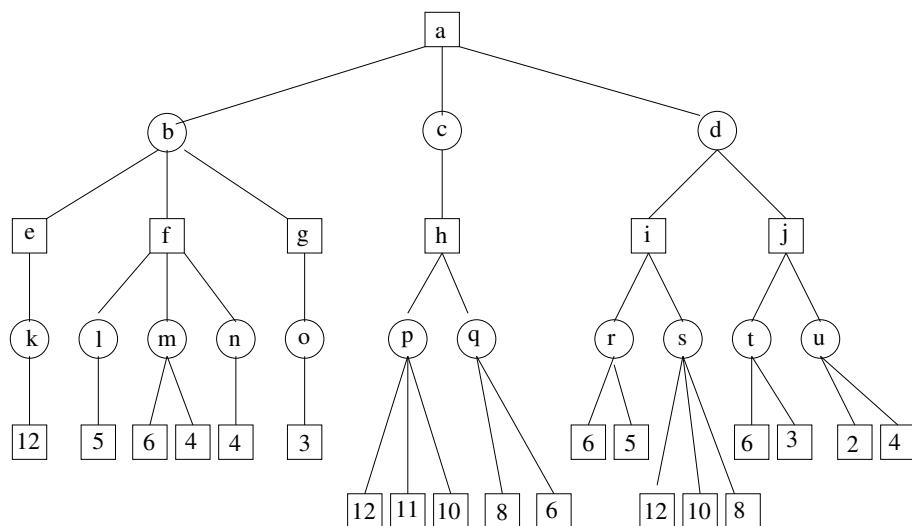
- (a) Se plantea solucionarlo mediante Hill-Climbing. Como solución inicial se parte de la solución vacía. Como operadores se usa el colocar un centro de acogida en un local vacío y quitar un centro de acogida de un local. Como función heurística se usa la suma de las distancias de los locales de cada uno de los  $C$  centros de acogida al resto, multiplicado por la suma de los voluntarios disponibles en cada local asignado.
- (b) Se plantea solucionarlo mediante Hill-Climbing utilizando como solución inicial el colocar aleatoriamente los  $C$  centros de acogida y usando como operadores de búsqueda mover un centro de acogida al local vacío más cercano respecto a su posición actual. Como función heurística se usa la suma de las distancias de los locales de cada uno de los  $C$  centros de acogida al resto más la suma de los voluntarios cercanos a cada local asignado.
- (c) Se plantea resolverlo mediante algoritmos genéticos. Para representar el problema utilizamos una tira de  $L$  bits, donde un bit a 0 indica que ese local no tiene centro de acogida asignado y un bit a 1 indica que si se le ha asignado un centro de acogida. Como población inicial generamos aleatoriamente  $n$  individuos donde en cada uno hay exactamente  $C$  bits a 1. La función heurística

es la suma de distancias de cada centro de acogida al resto más una constante  $k$  por la suma de los voluntarios disponibles de cada local asignado. Como operadores usamos un operador de cruce que intercambia aleatoriamente la mitad de los bits de cada individuo de la pareja cruzada y un operador de mutación que intercambia aleatoriamente dos posiciones de la tira de bits cumpliendo que en una sea un bit a 1 y en la otra un bit a 0.



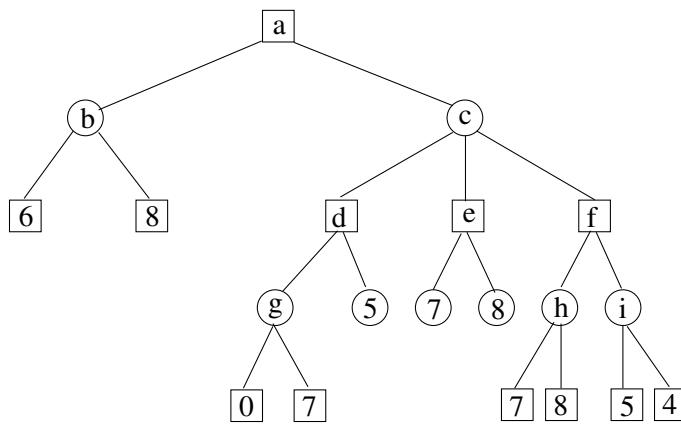
## 4. Jocs

- La figura següent mostra l'arbre d'un determinat joc, generat fins a una profunditat màxima de 4. Els números que acompañen als nodes del nivell 4 indiquen el valor de la funció d'avaluació estàtica per cadascun d'ells.

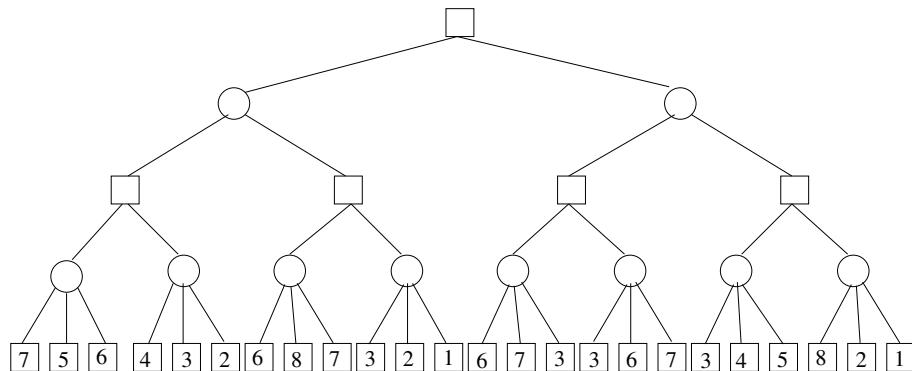


Contesta les següents qüestions:

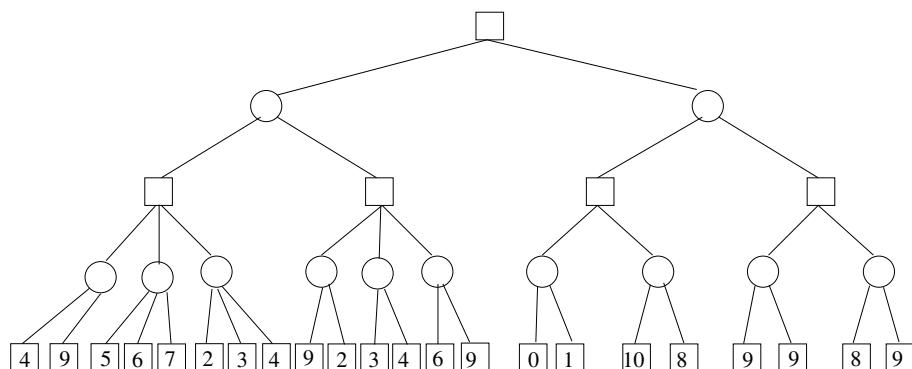
- (a) Seguint el mètode del Minimax, quin seria el valor assignat a cada node de l'arbre?  
(b) Aplica el mètode Minimax amb poda Alfa-Beta, usant els valors donats a la figura. Mostra el procés seguit, indicant quines són les branques podades i els valors assignats progressivament als nodes (tant els provisionals com el final).
- Donat aquest arbre on els nodes que mostren un número corresponen a l'avaluació estàtica de l'estat corresponent



- (a) Apliqueu la poda alfa-beta. Primer d'esquerra a dreta, i després de dreta a esquerra  
 (b) Com canvia el nombre de camins podats? Perquè?  
 (c) Sota quines condicions es podrien ordenar els nodes abans de començar la poda alfa-beta?
3. Aplica el minimax amb poda alfa-beta al següent arbre. Per a cada node explícita l'evolució dels valors alfa i beta. Indica clarament les podes i si es tracta d'una poda alfa o beta.



4. La siguiente figura muestra el árbol de búsqueda de profundidad 4 para un determinado juego para el jugador MAX. Los valores de los nodos terminales indican el valor de la función de evaluación para la posición obtenida en dicho nodo.



- (a) Utiliza el algoritmo Mínimax para decidir qué jugada debe escoger MAX, indicando claramente los valores de la función de evaluación propagados a cada nodo.  
 (b) Repite la exploración del apartado anterior utilizando la poda alfa-beta. Indica claramente la evolución de los valores de alfa y beta y los nodos que se podan.  
 (c) ¿Crees que modificando el orden de generación de nodos podría podarse un mayor número de nodos? Justifica brevemente la respuesta.

5. Donada la següent posició del joc de dames en un tauler de 4 per 4 on nosaltres juguem amb les peces rodones i toca jugar al contrari.

		3		4
	(1)			
			(2)	

Indica quina seria la valoració que min-max donaria a la jugada del contrari consistent en avançar la peça 3 cap a l'esquerra (en relació a la nostra posició davant el tauler), tenint en compte que:

- la valoració de cada posició es fa sumant peces pròpies i restant peces contràries, en base als següents valors individuals de les peces: 1 peça = 1 punt, 1 dama = 5 punts
- només cal cercar fins a una fondària de 4 jugades (a comptar des de la posició del dibuix). Cal que:
  - utilitzis l'algorisme min-max amb poda alfa-beta
  - generis els successors d'un estat seleccionant :
  - primer les peces etiquetades amb un nombre mes baix
  - primer el moviment cap a l'esquerra (en relació a la nostra posició davant el tauler).
  - dibuixis l'arbre generat indicant, per cada node, l'ordre de visita, el valor min-max calculat, l'evolució dels valors de alfa i beta i els valors propagats a d'altres nodes.

Nota per a aquells que NO saben jugar a dames:

- Moviment: Una peça es pot moure a qualsevol de les caselles adjacents, diagonals del davant (en relació a la posició pròpia davant del tauler).
- Una peça pot matar una peça contrària si aquesta està al davant i pot saltar pel damunt en diagonal (o sigui la posició on va a parar està lliure).
- Coronació: Una peça esdevé dama si es situa en una casella de la darrera fila (en relació a la posició pròpia davant del tauler).
- Moviment de dama: Una dama es pot desplaçar en qualsevol sentit de qualsevol diagonal i en un nombre no restringit de caselles.

6. Supongamos la siguiente posición en un juego de damas en un tablero 4 x 4. Las piezas propias son 1 y 2, las del contrario son 3 y 4. Acabo de jugar y la posición es:

4			
	3		
		(2)	
			(1)

Valorar la posición utilizando min-max con profundidad de búsqueda 4 (dos jugadas del contrario y dos propias) a partir de la posición del dibujo. La valoración de una posición se obtiene sumando los puntos de las pieza propias y restando las del adversario. Cada pieza vale 1 punto. Al coronar las piezas se convierten en damas y valen 5 puntos.

El orden de generación de los sucesores es:

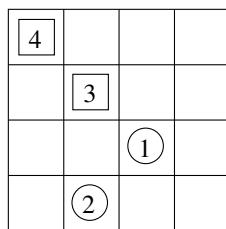
- la pieza 1 (3) tiene prioridad sobre la 2 (4)
- Si una pieza tiene más de un movimiento, tiene prioridad el de menor abscisa.

NO se aplicará la regla de la "bufada".

Se pide:

- Utilizar el algoritmo min-max SIN poda dibujando el árbol e indicando para cada nodo el orden de visita y el valor de la posición.
- Hacer lo mismo con el algoritmo min-max CON poda alfa-beta indicando además la evolución de los valores alfa/beta.
- Comparar los resultados, ¿ha habido ahorro en el número de nodos visitados? Nota per a aquells que NO saben jugar a dames veure regles al problema 5.

7. Supongamos la siguiente posición en un juego de damas en un tablero 4 x 4. Las piezas propias son 1 y 2, las del contrario son 3 y 4. Acabo de jugar y la posición es:



Valorar la posición utilizando min-max con profundidad de búsqueda 4 (dos jugadas del contrario y dos propias) a partir de la posición del dibujo. La valoración de una posición se obtiene sumando los puntos de las piezas propias y restando las del adversario. Cada pieza vale 1 punto. Al coronar las piezas se convierten en damas y valen 5 puntos.

El orden de generación de los sucesores es:

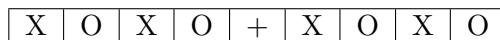
- la pieza que pueda comer. Si hubiera más de una, el orden sería (1, 2) o (3, 4)
- la pieza 1 (3) tiene prioridad sobre la 2 (4)
- Si una pieza tiene más de un movimiento, tiene prioridad el de menor abscisa.

NO se aplicará la regla de la "bufada".

Se pide:

- Aplicar el algoritmo min-max SIN poda, dibujando el árbol e indicando para cada nodo el orden de visita y el valor de la posición.
- Hacer lo mismo con el algoritmo min-max CON poda alfa-beta indicando además la evolución de los valores alfa/beta.
- Comparar los resultados, ¿ha habido ahorro en el número de nodos visitados? Nota per a aquells que NO saben jugar a dames veure regles al problema 5.

8. Tenemos el tablero que aparece en la figura, en la que el jugador MAX tiene las fichas marcadas como X y el min las fichas marcadas como O. La única ficha que pueden mover los jugadores es la marcada como +. Esta ficha se puede mover hacia la derecha y hacia la izquierda y tiene el efecto de cambiarse con la ficha que ocupa la posición a la que se desplaza y cambiar el signo de esa ficha y de la que pasa a ser su contigua. Por ejemplo, en la configuración XOXO+XOXO, si desplazamos la ficha + hacia la derecha obtenemos la configuración XOXOO+XXO. El objetivo de cada jugador es tener 5 fichas del mismo tipo seguidas.



Para la evaluación de las configuraciones utilizaremos la siguiente función:

$f'(n) = \text{Suma de los tamaños de los grupos de } X \text{ mayores que } 1 - \text{Suma de los tamaños de los grupos de } O \text{ mayores que } 1.$

Consideraremos que si la ficha + está entre dos fichas iguales, éstas no forman un grupo. Por ejemplo la evaluación de la configuración XOXOO+XXO sería  $(2 - 2) = 0$ , la evaluación de la configuración XOOOXO+OO sería  $(0 - 3 - 2) = -5$ .

- (a) Utiliza el algoritmo minimax con poda alfa-beta para evaluar el primer movimiento que debería hacer el jugador MAX, haz la exploración hasta el nivel 4 (dos jugadas de MAX y dos de MIN). Aplica siempre el mismo orden: primero el movimiento hacia la izquierda y después el movimiento hacia la derecha. Para cada nodo muestra claramente la evolución de los valores alfa y beta. ¿Cuál es el movimiento que debería escoger MAX?
  - (b) ¿Cuántos nodos nos ahorraremos respecto al uso del algoritmo sin poda?
9. El juego del Othelo se juega en un tablero de NxN como el de la figura y consiste en cubrir todo el tablero con fichas, colocándolas alternativamente, de manera que gane el que más fichas de su color consiga. Las reglas del juego son las siguientes: Para colocar una ficha hay que tener en esa fila, columna o diagonal otra ficha propia que encierre fichas contrarias con la que colocamos. Al colocar la ficha, todas las fichas de color contrario que queden atrapadas entre dos fichas propias cambian su color al nuestro.

El tablero comienza con las fichas que aparecen en la figura.

	N	B	
	B	N	
		B	N

- (a) Utiliza el algoritmo de minimax con poda alfa-beta para averiguar qué movimiento deberían hacer las blancas desde la posición inicial. Explora hasta profundidad 2 (un movimiento de las blancas y otro de las negras). Para la evaluación de los estados utiliza la siguiente tabla para asignar valor a cada ficha, el valor de un estado se obtiene sumando los valores de las fichas propias y restando las del contrario.

4	2	2	4
2	1	1	2
2	1	1	2
4	2	2	4

#### Orden de generación de sucesores:

Para expandir los nodos supón que el tablero está numerado desde la esquina superior izquierda siguiendo el orden por filas. Expande los nodos siguiendo el orden de esta numeración.

- (b) Utiliza el algoritmo MiniMax con poda alfa-beta para averiguar qué movimiento deberían hacer las negras desde la posición escogida por las blancas en el apartado anterior (en caso de empate escoge el primer movimiento que desarrollaste). Explora hasta profundidad 2 (un movimiento de las negras y otro de las blancas). Para la evaluación de las posiciones utiliza la tabla anterior.
10. Tenemos un tablero de 2x2 casillas. A cada casilla le asignamos un valor booleano (cierto/falso). El juego consiste en lo siguiente: dos jugadores se turnan en seleccionar una casilla del tablero. La casilla seleccionada queda eliminada de la partida y modifica el valor de la otra casilla de la misma fila y el valor de la otra casilla de la misma columna. A la casilla de la misma fila se le asigna el resultado de hacer la O lógica entre el valor que contiene y el de la casilla seleccionada. A la casilla de la misma columna se le asigna el resultado de hacer la Y lógica entre el valor que contiene y el de la casilla seleccionada. El juego termina cuando sólo queda una casilla. Empezando moviendo nosotros, ganamos la partida si la última casilla sin seleccionar tiene valor Falso y perdemos si queda con valor Cierto. A partir del siguiente tablero:

C	F
F	C

- (a) Aplica el algoritmo MiniMax para decidir qué casilla debemos seleccionar primero.
- (b) Repite la exploración aplicando poda alfa-beta.

Orden de generación de sucesores: empieza por la casilla de la esquina superior izquierda y sigue el sentido de las agujas del reloj.

11. El juego del reversi se juega en un tablero de NxN como el de la figura y consiste en cubrir todo el tablero con fichas, colocándolas alternativamente, de manera que gane el que más fichas de su color consiga. Las reglas del juego son las siguientes: Para colocar una ficha hay que tener en esa fila, columna o diagonal otra ficha propia que encierre fichas contrarias con la que colocamos. Al colocar la ficha, todas las fichas de color contrario que queden atrapadas entre dos fichas propias cambian su color al nuestro. El tablero comienza con las fichas que aparecen en la figura.

	N	B	
	B	N	

- (a) Utiliza el algoritmo de minimax con poda alfa-beta para averiguar qué movimiento deberían hacer las blancas desde la posición inicial. Explora hasta profundidad 2 (un movimiento de las blancas y otro de las negras). Para la evaluación de los estados utiliza la siguiente tabla para asignar valor a cada ficha, el valor de un estado se obtiene sumando los valores de las fichas propias y restando las del contrario.

4	2	2	4
2	1	1	2
2	1	1	2
4	2	2	4

Orden de generación de sucesores:

Para expandir los nodos supón que el tablero está numerado desde la esquina superior izquierda siguiendo el orden por filas. Expande los nodos siguiendo el orden de esta numeración.

- (b) Utiliza el algoritmo MiniMax con poda alfa-beta para averiguar qué movimiento deberían hacer las negras desde la posición escogida por las blancas en el apartado anterior (en caso de empate escoge el primer movimiento que desarrollaste). Explora hasta profundidad 2 (un movimiento de las negras y otro de las blancas). Para la evaluación de las posiciones utiliza la tabla anterior.
12. Tenemos el juego que muestra la figura, que consiste en lo siguiente:

1	2
2	1
1	2
2	1

Cada jugador posee una mitad del tablero, el jugador MAX tiene la mitad superior y el jugador MIN la mitad inferior. Cada jugador escoge en cada turno una de las celdas de su mitad. Cuando se escoge una celda, ésta pasa su valor a sus celdas adyacentes de la siguiente manera: 2 puntos (o 1 si la celda sólo contiene 1) pasan a la celda inferior en el caso de MAX o superior en el caso de MIN, el resto de puntos que quedan se reparten, primero un punto a la celda de la izquierda y después un punto a la celda de la derecha, si aun quedan puntos, uno pasa a la celda superior en el caso de MAX o inferior en el caso de MIN. Los puntos que quedan se mantienen en la celda elegida. Si se reparten todos los puntos la celda queda con un valor de 0. Se gana cuando se consigue que en la zona del contrario la suma de los valores de las celdas sea igual o superior a 9.

- (a) Aplicar el algoritmo minimax con poda alfa-beta dos niveles (una jugada de MAX y otra de MIN) para averiguar cual sería la jugada de MAX a partir de la posición inicial. Para evaluar las posiciones se ha de utilizar la siguiente función:

$$\text{Vposición} = \text{VMAX} - \text{VMIN}$$

$\text{VMAX} =$  suma de valores de la zona MIN - suma de diferencias de la zona MAX (valor de celda mayor de 2 - 2)

$\text{VMIN} =$  suma de valores de la zona MAX - suma de diferencias de la zona MIN (valor de celda mayor de 2 - 2)

Por ejemplo:

0	2
3	3
1	0
2	1

$$\text{VMAX} = 4 - (1+1) = 2, \text{VMIN} = 8 - 0 = 8, \text{Vposición} = -6$$

Para elegir el orden de expansión de los nodos, comenzar por la celda de mayor valor. En caso de empate se escoge la que esté más cerca de la mitad y, en caso de nuevo empate, de izquierda a derecha.

- (b) A partir de la jugada elegida por MAX, hacer una nueva búsqueda para averiguar cual será la respuesta de MIN.

13. El siguiente juego sobre un tablero 3x4 empieza con las fichas 'X' en la fila 1 y con las fichas 'O' en la fila 4. Un jugador gana cuando consigue colocar todas sus fichas en la fila del contrincante. Dada la siguiente situación del juego, donde le toca jugar al contrincante (fichas 'O'),

	1	2	3
1	X		
2	X		
3	O	X	O
4			O

teniendo en cuenta que:

- Las fichas 'X' avanzan sólo hacia abajo, ya sea en diagonal o en vertical.
- Las fichas 'O' sólo avanzan hacia arriba, de la misma manera.
- Las fichas sólo pueden avanzar una posición en cada turno y siempre a una posición libre. Sin embargo, una ficha puede saltar sobre una o dos fichas contrarias contiguas.

y aplicando los movimientos ESTRICAMENTE en el siguiente orden. Selección de fichas: mover primero la ficha más cercana al objetivo que esté más a la izquierda, seleccionar la segunda con el mismo criterio y luego mover la tercera. Orden de selección de movimientos: 1) Saltar. 2) Mover a diagonal izquierda. 3) Mover en vertical. 4) Mover a diagonal derecha.

- (a) Aplicar el algoritmo minimax con una profundidad máxima 2 para obtener la valoración de la configuración de la figura. Utiliza la siguiente función de evaluación de estados:  $\text{FAE} = \text{V('O')} - \text{V('X')}$  donde  $\text{V}(*)$  es el número de movimientos posibles para las fichas \*. Si no hay posibilidad de movimientos para esas fichas, el valor de  $\text{V}(*)$  será  $\infty$ .
- (b) Aplicar el algoritmo minimax  $\alpha\beta$  bajo las mismas circunstancias para el mismo propósito.
- (c) ¿Cuántos nodos se han podado?

- (d) ¿Existe algún orden de aplicación de movimientos diferente al exigido que podría provocar más podas?
14. El siguiente juego sobre un tablero de 3x4 empieza con las fichas 'A' en la fila 1 y con las fichas 'B' en la fila 4. Un jugador gana cuando consigue colocar todas sus fichas en la fila del contrincante. Dada la siguiente situación del juego, donde el contrincante acaba de jugar (fichas 'B'),

	1	2	3
1	A		
2	A	B	A
3			B
4			B

teniendo en cuenta que:

- Las fichas 'A' avanzan sólo hacia abajo, ya sea en diagonal o en vertical.
- Las fichas 'B' sólo avanzan hacia arriba, de la misma manera.
- Las fichas sólo pueden avanzar una posición en cada turno y siempre a una posición libre. Sin embargo, una ficha puede saltar una o dos fichas contrarias contiguas.

y aplicando los movimientos ESTRICAMENTE en el siguiente orden. Selección de fichas: mover primero la ficha más cercana al objetivo que esté más a la izquierda, seleccionar la segunda con el mismo criterio y luego mover la tercera. Orden de selección de movimientos: 1) Saltar. 2) Mover a diagonal izquierda. 3) Mover en vertical. 4) Mover a diagonal derecha.

- (a) Aplicar el algoritmo minimax con una profundidad máxima 2 para obtener la siguiente jugada a realizar. Utiliza la siguiente función de evaluación de estados:
- $$FAE = V('B') - V('A')$$
- donde  $V('*')$  es el número de movimientos posibles para las fichas \*. Si no hay posibilidad de movimientos para esas fichas, el valor de  $V('*')$  será  $\infty$ .
- (b) Aplicar el algoritmo minimax  $\alpha\beta$  bajo las mismas circunstancias para el mismo propósito. ¿Qué jugada te recomienda ahora?
- (c) ¿Cuántos nodos se han podado?
- (d) ¿Existe algún orden de aplicación de movimientos diferente al exigido que podría provocar más podas?

15. Tenemos el juego que muestra la figura, que consiste en lo siguiente:

$$1111+1111$$

la única pieza móvil es el + y se puede mover a la derecha o a la izquierda, el efecto que tiene es intercambiarse con el número de la dirección a la que se desplaza y sumar a todas las posiciones que están en el lado del movimiento el número con el que se intercambia. Por ejemplo:

$$\begin{aligned} &1111+1111 \\ &\text{(Desplazamos + a la izquierda)} \\ &222+11111 \\ &\text{(Desplazamos + a la izquierda)} \\ &44+211111 \end{aligned}$$

Tomando como referencia todo el tablero y no importando donde está el +, el jugador MAX posee los cuatro números de la izquierda y el MIN los cuatro números de la derecha. Gana el jugador que consigue que sus números sumen más de 50. Aplicar el algoritmo minimax con poda alfa-beta explorando cuatro niveles (dos jugadas de MAX y otras dos de MIN) para averiguar cual sería la jugada de MAX a partir de la posición inicial. Para evaluar las posiciones se ha de utilizar la siguiente función:

$$V_{\text{posición}} = \text{Suma de los números de MAX} - \text{suma de los números de MIN}$$

El orden de expansión es primero los movimientos de la izquierda y después los de la derecha.

16. Una persona está jugando al Conecta-4 en su ordenador. Desde el punto de vista de la máquina, su contrincante es la persona. El juego consta de un tablero de 4 pilas (columnas) de altura 4 (filas). Cada jugador dispone de 8 fichas ('X' para el ordenador y 'O' para la persona). En cada turno de un jugador, éste puede colocar una de sus fichas en una de las 4 pilas siempre y cuando la pila contenga menos de 4 fichas. Una vez colocada una ficha, ésta no se puede volver a mover.

Un jugador gana cuando consigue que 4 de sus fichas queden alineadas en horizontal, en vertical o en diagonal. Suponiendo el siguiente estado del juego en el que le toca jugar al ordenador (X):

O			
X		O	X
X	O	X	O
1	2	3	4

- (a) ¿Cuál sería el movimiento que haría la máquina si el juego estuviera implementado con un algoritmo minimax de profundidad 2? Representa los pasos seguidos por dicho algoritmo suponiendo que se está utilizando la siguiente FAE:

FAE = posibles alineaciones para el ordenador - posibles alineaciones para la persona  
donde una posible alineación significa la existencia de una posibilidad de llegar a conseguir una alineación en lo que queda de juego. Por ejemplo, en el estado dibujado, la persona (O) tiene 4 posibles alineaciones (una en diagonal, una vertical y dos horizontales) mientras que el ordenador (X) tiene dos posibilidades (una en diagonal y una horizontal).

**ORDEN ESTRICTO:** Considera las posibles jugadas recorriendo las columnas de izquierda a derecha.

- (b) ¿Cuál sería dicho movimiento si la implementación incorporase podas  $\alpha\beta$ ? Representa los pasos seguidos por dicho algoritmo de forma separada del apartado (a). Para ello etiqueta con letras mayúsculas (A, ..., Z, AA, ...) cada estado del apartado (a) y utilízalas aquí. ¿Cuántos nodos se han podado?

17. El juego del Nim se juega a partir de varias filas de palillos de las cuales un jugador puede retirar, en un turno, el número que desee de palillos pero de una única fila. El jugador que retira el último palillo pierde.

Podemos representar la configuración de las filas mediante secuencias de enteros. Por ejemplo, (1, 3, 5) indica que hay tres filas con uno, tres y cinco palillos respectivamente.

Se pide:

- (a) A partir de la configuración (1, 2, 2), utiliza el algoritmo minimax para averiguar qué movimiento deberíamos realizar primero.  
(b) Repite la exploración utilizando la poda alfa-beta. ¿Qué movimiento nos aconseja? ¿Cuántos nodos se han podado?

Orden de generación de sucesores: de izquierda a derecha según el orden indicado en la secuencia de enteros y aplicando todos los movimientos posibles para cada fila antes de pasar a la siguiente. Dentro de cada fila, se comenzará por retirar un palillo, luego dos, y así sucesivamente. Si el estado resultante de la aplicación de un movimiento es una permutación de otro estado hermano anterior, no hay que incluirlo en el árbol (por ejemplo, de (1,2,2) se puede pasar a (1,1,2) y también a (1,2,1), pero este último no se incluirá en el árbol). Estados finales: En el árbol, son estados finales tanto los estados con cero palillos como los estados con un palillo ya que en ambos casos podemos marcar quien es el ganador.

18. Tenemos el tablero que aparece en la figura, en la que el jugador MAX tiene las fichas marcadas como A y el MIN las fichas marcadas como B. Los jugadores pueden mover sus fichas y las del contrario. Los movimientos posibles son dos:

- desplazamiento de una ficha a la casilla libre contigua
- salto de una ficha sobre otra contraria (sólo una) cambiando de signo la ficha contraria, siempre y cuando la siguiente casilla esté libre. Ejemplo: ABA\_BAB  $\Rightarrow$  A\_BBBAB

Restricción: una jugada de desplazamiento no es válida si simplemente está deshaciendo la jugada anterior. Por ejemplo, si un jugador pasa de AB\_BBAB al estado ABB\_BAB, el otro jugador no puede volver a pasar a AB\_BBAB. El objetivo de cada jugador es tener 4 fichas del mismo tipo estrictamente consecutivas.

A	B	A		B	A	B
---	---	---	--	---	---	---

Para la evaluación de las configuraciones utilizaremos la siguiente función:

$f'(n)$  = tamaño del mayor grupo de As consecutivas - tamaño del mayor grupo de Bs consecutivas.

Ejemplos:

$$f'(ABA\_BAB) = 1 - 1 = 0$$

$$f'(ABB\_BAB) = 1 - 2 = -1$$

- Utiliza el algoritmo minimax con poda alfa-beta para evaluar cual debería ser el primer movimiento del jugador MAX. Haz la exploración hasta el nivel 3 (dos jugadas de MAX y una de MIN). Aplica siempre el mismo orden: movimientos posibles recorriendo el tablero de izquierda a derecha. Para cada nodo muestra claramente la evolución de los valores alfa y beta. ¿Cuál es el movimiento que debería escoger MAX?
- ¿Cuántos nodos nos ahorramos respecto al uso del algoritmo sin poda?



## 5. Satisfacció de restriccions

1. Una empresa de alquiler de vehículos dispone de una flota con las siguientes características:

	Marca	Color	Precio/día
Coche1	Ford	Blanco	30 euros
Coche2	Citroen	Azul	35 euros
Coche3	Mercedes	Gris	60 euros
Coche4	Citroen	Verde	30 euros
Coche5	Seat	Rojo	35 euros
Coche6	Opel	Blanco	35 euros
Coche7	Mercedes	Rojo	35 euros
Coche8	BMW	Negro	60 euros
Coche9	Citroen	Gris	40 euros
Coche10	Seat	Azul	40 euros

La empresa ha recibido la siguiente serie de peticiones que ha de intentar satisfacer:

	Precio/día	Color
Petición1	$\leq 40$ euros	NO Azul
Petición2	$\leq 30$ euros	Azul O Blanco
Petición3	$\leq 60$ euros	NO Gris y NO Negro
Petición4	$\leq 40$ euros	NO Rojo y NO Azul
Petición5	$\leq 60$ euros	Blanco

Tenemos la restricción adicional de que no puede haber dos coches de la misma marca. Considera las peticiones como variables, en el mismo orden en que figuran, y los coches de la flota como valores, también en el mismo orden en que se listan.

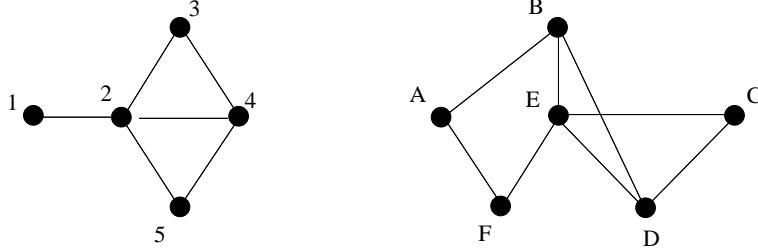
- (a) Aplica forward-checking para resolver el problema. Desarrolla el proceso hasta el punto en el que encuentres una solución. Indica en cada paso sólo los dominios que se modifican.
- (b) Aplica backtracking cronológico hasta el segundo backtracking a la variable petición 1.

2. El director de la coral infantil "Veus suaus" tiene que decidir el orden de colocación de los ocho cantores. Debe distribuirlos en dos filas (A, B) de forma que en cada fila queden en orden decreciente de altura, colocando el más alto a la izquierda (posición 1). Además, la altura de cada niñ@ de la fila trasera (A) debe ser superior o igual a la del que tenga delante. Finalmente, no quiere colocar dos herman@s seguid@s en la misma fila ni un@ delante del otr@. La relación de niñ@s y sus alturas es la siguiente:

Nombre	Altura
Esteva Blanco (EB)	1,40
Pedro Costa (PC)	1,60
Ana Costa (AC)	1,50
Juan Costa (JC)	1,30
Oriol Pi (OP)	1,40
María Ruiz (MR)	1,60
Rosa Sánchez (RS)	1,50
Carla Sánchez (CS)	1,30

Considera como variables las posiciones de izquierda a derecha en las filas (A1, A2, A3, A4, B1, B2, B3, B4) y como valores las iniciales de los nombres de los cantores. El orden de recorrido de las variables es el indicado entre paréntesis. El orden de recorrido de los valores es el de la lista anterior.

- (a) Aplica forward-checking para resolver este problema. Desarrolla el proceso hasta el punto en el que hay que realizar el primer backtracking a B1.
- (b) Aplica búsqueda en profundidad y backtracking cronológico hasta el primer backtracking a A2.
3. Deseamos utilizar la técnica de satisfacción de restricciones para resolver el problema de comprobar si un grafo está contenido en otro. Dados los dos grafos siguientes:



Deseamos saber si el primer grafo está contenido en el segundo.

Para resolver el problema debemos asignar a cada nodo del primer grafo un nodo del segundo grafo de manera que las conexiones entre los nodos del primer grafo se respeten en el segundo grafo.

- (a) Define los dominios de cada variable (nodos del primer grafo) y aplica las restricciones que creas posibles en cada variable para reducir el número de valores posibles (las que sean evidentes por las características del problema, no apliques arco consistencia).
- (b) Resolver el problema aplicando el algoritmo del forward checking
- (c) Resolver el problema aplicando el algoritmo de backtracking cronológico hasta que la variable 1 tome el valor C
- (d) Si tomamos como variables los nodos del segundo grafo ¿cuales serían los valores de los dominios de estas variables?
4. Dado un vector de cinco posiciones se desea obtener una asignación de letras tal que no haya dos letras consecutivas iguales y que el conjunto sea capicúa.
- Los valores posibles para la posición 1 son A,B,C,D,E.
  - Los valores posibles para las posiciones 2 y 3 son A,B,C.

- Los valores posibles para la posición 4 son C,D,E.
- Los valores posibles para la posición 5 son B,C,D,E.

Ejemplos de asignaciones válidas: E-C-A-C-E     D-C-B-C-D

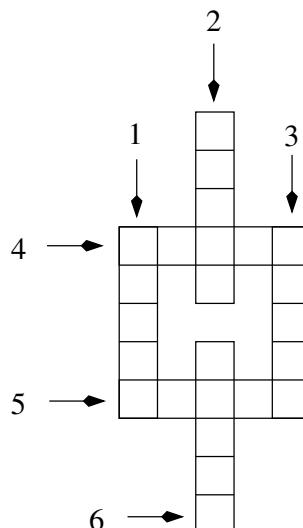
- Resolver el problema mediante backtracking cronológico.
- Resolver el problema mediante forward checking.

5. Como cada año, los protagonistas de los más famosos culebrones se reunen para celebrar la realización de 100 nuevos capítulos de su teleserie. Lamentablemente la rivalidad entre ellos es tal que algunos no se pueden sentar al lado de otros durante la cena de celebración. Nuestros protagonistas son: Carlos Miguel, David Miguel, Carlos David, Juan Miguel, Pedro Miguel, Juan Luis y Juan Carlos. Las restricciones son tales que un Carlos no se puede sentar junto a un Pedro, ni un Juan se puede sentar al lado de otro Juan.

- Utiliza el algoritmo de forward checking para dar una solución a la ubicación de estos 7 personajes en una mesa circular (cada comensal sólo tiene dos vecinos, uno a la derecha y otro a la izquierda). Usa para la exploración de los valores el orden en el que están en el enunciado.
- Resuelve el mismo problema aplicando el algoritmo de backtracking cronológico.

6. Deseamos resolver el crucigrama de la figura colocando las siguientes palabras: ROLLO, SALVO, SOLAR, ROCAS, OCIOS, SILOS.

- Para resolverlo aplica el algoritmo de forward checking utilizando las posiciones para las palabras como variables y las palabras como valores utilizando la numeración para las variables que se indica la figura y explorando las palabras en el orden en que se dan en el enunciado.
- Resuelve ahora el problema aplicando el backtracking cronológico.



NOTA: las palabras horizontales se colocan de izquierda a derecha y las palabras verticales de arriba a abajo. No hay palabras escritas al revés.

7. Una petita empresa ha de comprar telèfons mòbils pels seus directius. Han estat mirant preus de companyies telefòniques i es troben amb la següent informació:

Companyia	Tarifa
MultiStaf (MS)	5000
AguaTel (AT)	6000
FunkyTel (FT)	4500
TopeVisión (TV)	3500

Aquesta petita companyia té cinc executius i vol assignar-los un telèfon mòbil a cadascun sota les següents restriccions:

- No pot haver-hi més de dos executius amb la mateixa companyia.
- El cost total de l'assignació de companyies a executius no ha de superar les 21.000 ptes.

Es demana:

- (a) Resoldre completament el problema aplicant forward checking.
- (b) Desenvolupar l'aplicació del backtracking cronològic fins al punt en que el backtracking ens porta a revisar la primera assignació feta a la segona variable.

NOTA: Per ambdós apartats considereu que els executius es representen per E1, E2, E3, E4 i E5. Els valors possibles són MS, AT, FT, TV i cal considerar-los sempre en aquest ordre.

8. La compañía de aviación “Air Vostrum” debe realizar habitualmente la tarea de configurar la tripulación de los vuelos. El problema actual consiste en organizar parejas de comandante y piloto para cubrir cuatro vuelos: París, Roma, Beijing y Tokio. El personal disponible es:

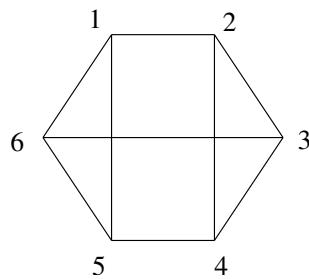
Comandantes			Pilotos		
C1	Pérez	45 años	P1	Asensio	32 años
C2	Benitez	43 años	P2	Martín	35 años
C3	Almansa	40 años	P3	Marín	38 años
C4	Morales	47 años	P4	Casales	40 años

Las normas de la compañía, para vuelos fuera de Europa, impiden que la suma de edades del comandante y el piloto exceda de 75 años. Adicionalmente, los comandantes son muy supersticiosos y no admiten que su piloto tenga como inicial de apellido la misma que ellos. Aplica el algoritmo de forward checking para configurar las cuatro tripulaciones necesarias. Explica los valores posibles de cada variable en cada paso. Utiliza variables y valores en el orden en que aparecen en el enunciado.

9. El senyors Grífol tenen el problema habitual de cada Nadal i que consisteix en comprar regals per als seus nebots/nebodes. Han de comprar regals per als germans Joan i Maria, per als germans Pere, Ana i Oriol i per en Ramon i en Xavier, ambdós fills únics. Han decidit que compraran dos jocs iguals d'escacs, dos jocs iguals d'experiments de química i tres llibres didàctics iguals. Per a repartir aquests regals entre els nebots han de tenir en compte que, per raons òbvies, no poden donar el mateix regal entre germans, que ni el Joan ni el Xavier poden tenir el mateix regal que l'Oriol i que el Ramon no pot tenir el mateix regal que l'Ana.

NOTA: Utilitza els noms dels nebots en l'ordre en que apareixen a l'enunciat. Els regals s'identifiquen per E1, E2, Q1, Q2, L1, L2, L3 i també cal respectar aquest ordre.

- (a) Dibuixa el graf de restriccions entre les variables.
  - (b) Resol el problema aplicant forward checking. Indica clarament a l'inici i a cada pas el domini de les variables.
  - (c) Inicia el procés de resolució aplicant cerca en profunditat i backtracking cronològic. Desenvolupa el procés fins al moment en que es produeix el segon backtracking cap a Oriol.
  - (d) Creus que el backtracking cronològic trobarà la solució? Justifica la resposta.
10. Una empresa de telefonía móvil quiere colocar seis antenas en una ciudad cuyas posibles ubicaciones están reflejadas por este grafo:



Donde cada arco indica que existe visión directa entre dos posiciones. Las antenas tienen las características siguientes:

Antena	Frecuencia	Potencia
A	20 GHz	1 Mw
B	1.8 GHz	2 Mw
C	20 GHz	3 Mw
D	1.8 GHz	1 Mw
E	20 GHz	1 Mw
F	1.8 GHz	2 Mw

Las restricciones de colocación de las antenas son las siguientes: No se pueden colocar dos antenas consecutivas de la misma frecuencia. No puede haber dos antenas con visibilidad directa cuya suma de potencia sea superior a 4 Mw.

- (a) Considerando las posiciones de las antenas como variables y las antenas como los dominios de estas variables, utiliza el algoritmo del forward checking para buscar una solución a la colocación de las antenas usando para la exploración el orden en el que aparecen las posiciones y las antenas en el enunciado. Desarrolla el algoritmo hasta el primer backtracking a la primera variable.
  - (b) Desarrolla el algoritmo de bactracking cronológico hasta el punto en que deba realizarse el primer backtracking. Para cada asignación que falle indica brevemente el motivo.
11. El responsable del periódico "Noticias frescas" debe confeccionar la primera página de la edición de mañana que consta de cuatro posiciones tal como se ve en la figura

Noticias Frescas 8-junio-2002	
P1	P2
P3	P4

Dispone de siete noticias con las siguientes características:

	Ámbito	Tema
N1:	Nacional	Política
N2:	Nacional	Sucesos
N3:	Internacional	Política
N4:	Nacional	Deportes
N5:	Nacional	Política
N6:	Internacional	Sociedad
N7:	Internacional	Sucesos

Las normas de redacción del periódico imponen las siguientes restricciones:

- Una noticia internacional sólo puede estar en una posición inferior (P3 o P4) si la de justo encima es también internacional.
- Sólo puede haber dos noticias del mismo tema si están en diagonal.
- Una noticia de política y otra de sucesos no pueden estar en la misma horizontal.

Considerando las variables P1..P4 y los valores N1..N7 y respetando el orden numérico, se pide:

- (a) resolver el problema de la confección de la primera página aplicando forward-checking
- (b) resolver el mismo problema mediante backtracking cronológico.

12. Dos cadenas de televisión desean coordinar sus franjas horarias de manera que el telespectador tenga más donde elegir. Supondremos que cada cadena divide su horario en cuatro franjas, y que en cada franja podemos tener uno de estos tres tipos de programa: Fútbol, Concurso o Película. Tenemos como restricciones:

- No puede haber simultáneamente el mismo tipo de programa en las dos cadenas.
- No puede haber en una misma cadena dos programas seguidos del mismo tipo.
- No puede haber en total más de dos partidos de fútbol o concursos.

Utilizando las franjas horarias como variables y siguiendo el orden: Fr1.1, Fr1.2, Fr1.3, Fr1.4, Fr2.1, Fr2.2, Fr2.3, Fr2.4 (donde el primer número es la cadena y el segundo la franja horaria) y usando para los valores posibles el orden fútbol, concurso, película, utiliza el algoritmo de forward checking para encontrar una solución a este problema.

13. Deseamos construir un circuito ubicando los componentes de manera que las distancias que los componentes permitan su correcto funcionamiento. El circuito sobre el que queremos trabajar es una cuadrícula de 3x3, de manera que cada componente tiene asignado unas coordenadas.

Tenemos que ubicar 4 componentes (A,B,C,D), las restricciones entre ellos son:

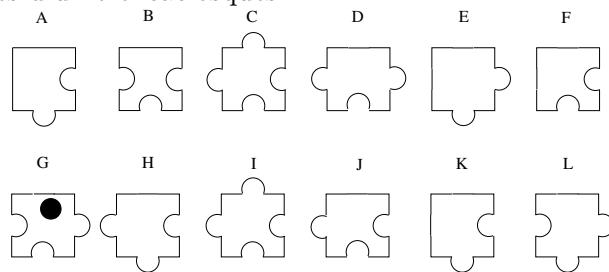
- La distancia entre el componente A y B ha de ser mayor o igual a 2
- La distancia entre el componente A y C ha de ser 1
- La distancia entre el componente B y C ha de ser 2
- La distancia entre el componente B y D ha de ser 1
- La distancia entre el componente C y D ha de ser 1

Para calcular la distancia entre dos componentes se usa la función:  $d(a,b)=|ax-bx|+|ay-by|-1$ , donde ax es el valor de la coordenada x del componente a, ay es el valor de la coordenada y del componente a, ídem para el componente b Considera los componentes como variables y las componentes como los valores de estas variables. Como orden de exploración, empieza por la esquina superior izquierda y sigue el orden de izquierda a derecha y de arriba a abajo.

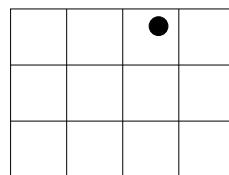
- (a) Utiliza el algoritmo de forward checking para explorar el problema hasta llegar al tercer valor de la variable A
- (b) Utiliza el algoritmo de backtracking cronológico para explorar el problema hasta el primer backtracking a la variable B
- (c) ¿Se puede simplificar el primer problema eliminando valores del dominio de las variables propagando las restricciones? ¿Porqué?
- (d) ¿Se puede aprovechar alguna característica del problema para simplificarlo y probar menos valores en la búsqueda?

- (e) En muchas ocasiones se pueden intercambiar variables y valores en los problemas de satisfacción de restricciones. ¿Cuáles serían los dominios de valores si usáramos como variables las posiciones en lugar de los componentes?

14. Tenim les següents peces d'un trencaclosques:



Les 12 peces han de formar la següent imatge:



Fent servir les següents posicions del trencaclosques com a variables i les peces com a valors (poden efectuar-se rotacions sobre les peces, excepte per la C i la I):

1	2	3	4
10	11	12	5
9	8	7	6

el domini de cada variable queda inicialment restringit, donades les característiques del puzzle.

- (a) Utilitza backtracking cronològic per completar el trencaclosques.
- (b) Fes-ho ara amb forward checking.

**NOTA:** En ambos casos, feu servir l'ordre numèric per les variables i l'ordre alfabètic pels valors. Deixa clar quins són els passos donats!!

15. Una empresa de consultoría tiene que organizar el trabajo de tres proyectos durante tres días y dispone de cuatro consultores para llevarlos a cabo. Cada consultor puede dedicar un conjunto de horas a cada proyecto, este conjunto de horas está dividido en fracciones (un consultor sólo tiene dos fracciones de dedicación al día), las horas de cada consultor en cada fracción de dedicación son las siguientes:

Consultor	Disponibilidad
C1	2.5
C2	4
C3	3
C4	1.5

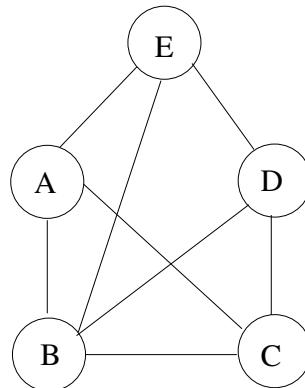
Existen las siguientes restricciones de asignación:

- Un consultor no puede trabajar dos días seguidos en el mismo proyecto
- Un consultor sólo puede dedicar dos fracciones de dedicación al día
- El número total de horas dedicadas a un proyecto ha de ser inferior o igual a 8
- Todos los días ha de haber algún consultor asignado a un proyecto

Utilizar los proyectos combinados con los días como variables, asignando valores para todos los proyectos para el primer día, para el segundo día y para el tercer día ( $P1d1, P2d1, P3d1, P1d2, \dots$ ), usa los consultores como valores ( $C1-C4$ )

- (a) Resolver el problema aplicando forward checking. En cada paso indica solamente las variables en las que hay algún cambio en su dominio.
  - (b) Describe que otras maneras hay de escoger variables y valores para hacer la exploración y que ventajas o inconvenientes tienen.
16. El club de natació “Dofins mulars” ha d’inscriure per a les properes competicions dos equips de relleus (A i B). Els nedadors a inscriure (amb les seves corresponents edats) són: Manel(10), Oriol(10), Adrià(11), Enric(11), Xavier(12), Pere(12), Joan(13) i Ramon(13). Cada equip està format per quatre nedadors. L’entrenador ha de tenir en compte les següents restriccions:
- La suma d’edats de cada equip no pot ser superior a 46.
  - No poden haver-hi en el mateix equip dos nedadors de 10 anys.
  - No poden haver-hi en el mateix equip dos nedadors de 13 anys.
  - El Pere i el Joan no poden estar en el mateix equip.
  - En Xavier i l’Adrià no poden estar en el mateix equip.
- Considera els nedadors com les variables del problema i tracta-les en l’ordre en que apareixen els noms a l’enunciat.
- (a) Representa les restriccions de la 2 a la 5 mitjançant un graf.
  - (b) Resol el problema aplicant forward checking. Indica a cada pas només els dominis que es modifiquen. Quan correspongui, indica breument per què cal assignar un nou valor o fer backtracking.
  - (c) Resol el problema aplicant cerca en profunditat amb backtracking cronològic.
17. Dado el siguiente grafo de restricciones donde cada restricción es una condición de desigualdad y los siguientes dominios para las variables:

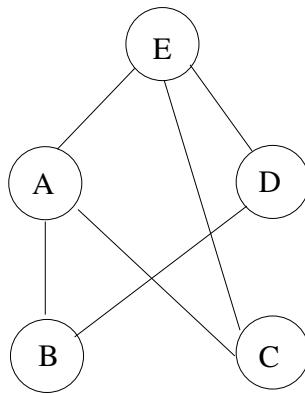
$$\begin{aligned} A &= \{1,2,3\} \\ B &= \{1,2\} \\ C &= \{2,3\} \\ D &= \{2,3\} \\ E &= \{1,2,3\} \end{aligned}$$



Haz la ejecución del backtracking cronológico hasta el primer backtracking a la variable A y del forward checking hasta encontrar la primera solución

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

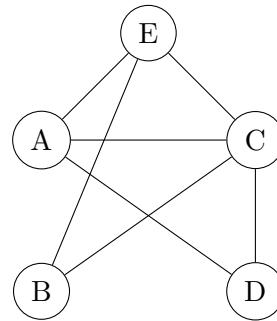
$$\begin{aligned} A &= \{1,2\} \\ B &= \{2,3\} \\ C &= \{1,3\} \\ D &= \{2,3\} \\ E &= \{1,3\} \end{aligned}$$



Haz la ejecución del forward checking hasta encontrar la primera solución

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

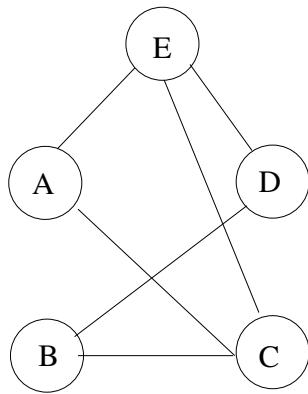
$$\begin{aligned} A &= \{1,2\} \\ B &= \{2,3\} \\ C &= \{1,2\} \\ D &= \{1,2,3\} \\ E &= \{1,2,3\} \end{aligned}$$



Haz la ejecución del forward checking hasta encontrar la primera solución

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

$$\begin{aligned} A &= \{1,2\} \\ B &= \{1,3\} \\ C &= \{1,2\} \\ D &= \{1,3\} \\ E &= \{1,2\} \end{aligned}$$



Haz la ejecución del forward checking hasta encontrar la primera solución



## 6. Anàlisi de mètodes de cerca

1. Deseamos construir un circuito integrado de manera que la intensidad de corriente necesaria por los diferentes elementos esté equilibrada en todo el circuito. Para simplificar el problema hemos supuesto que el circuito es una cuadrícula de  $N \times M$ . Los diferentes elementos que queremos colocar en el circuito son de  $K$  tipos diferentes y tenemos que colocar un número  $e_k$  de cada uno de ellos. Cada tipo de elemento necesita una intensidad de corriente específica.

Cada posición del circuito puede albergar  $P$  elementos de cualquier tipo y hemos de colocar todos los elementos.

El objetivo es conseguir que la suma de las intensidades que necesitan los elementos para cada fila y cada columna no supere un valor  $I$  y que la diferencia de intensidad de corriente entre una celda y cada una de sus cuatro vecinas contiguas no sea mayor que un valor  $V$ .

Se nos proponen las siguientes soluciones:

- (a) Usar el algoritmo de Hill Climbing. El estado incluye, para cada componente (de los  $C = \sum_{k \in K} e_k$  componentes), la coordenada en que se ubica, y estructuras adicionales que hagan falta para computar el resto de cosas de forma eficiente (como la intensidad total en una casilla según la asignación actual). Como solución inicial se construye una asignación vacía (toda pieza falta por asignar). Como operador usamos `colocar(pieza, fila, columna)`, que asigna la pieza a la posición  $(fila, columna)$ . La función heurística es:

$$h(n) = \sum_{i=1}^N sobrecarga\_fila(i) + \sum_{j=1}^M sobrecarga\_columna(j) + \sum_{i=1}^N \sum_{j=1}^M sobrecarga\_vecinas(i, j) + W$$

donde `sobrecarga_fila(i)` y `sobrecarga_fila(j)` valen 1 si la suma de intensidades de la fila o columna superan  $I$  respectivamente, `sobrecarga_vecinas(i, j)` cuenta el número de diferencias de intensidad superiores a  $V$  entre la celda  $(i, j)$  y sus cuatro celdas vecinas, y  $W$  es el número de piezas todavía sin colocar.

- (b) Usar un algoritmo de programación de restricciones. Las variables son los diferentes elementos a colocar y los valores son las coordenadas  $(i, j)$  de la cuadrícula donde se podrían colocar. Las restricciones consideradas son que la suma de las intensidades de los elementos que comparten la misma fila y los que comparten la misma columna sea menor que  $I$  y que la diferencia entre

la suma de las intensidades de los elementos de una celda con la suma de las intensidades de los elementos de las celdas vecinas contiguas no sea superior en  $V$ .

Comenta cada una de las soluciones que se proponen, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

2. Una empresa nos pide un sistema inteligente para acelerar la transmisión de datos entre máquinas. Queremos transmitir  $X$  Kbits entre dos máquinas  $M_I$  y  $M_F$ , para ello hemos de establecer  $C$  canales de comunicación que han de atravesar  $N$  máquinas ( $M_1, M_2, \dots, M_N$ ) que hacen de puntos intermedios ( $C \leq N$ ), los  $X$  Kbits los dividimos a partes iguales por cada canal. Conocemos la velocidad en  $kbps$  del canal que se puede establecer entre cada par de máquinas (incluidas  $M_I$  y  $M_F$ ), queremos que la velocidad media de un canal sea como mínimo  $V$  Kbps y que un canal no atraviese más de  $P$  máquinas. Buscamos la solución que conecte todas las máquinas sin que dos canales pasen por la misma máquina (exceptuando  $M_I$  y  $M_F$ ) tardando lo menos posible en transmitir los  $X$  Kbits.

Tras un análisis inicial del problema un compañero de nuestra empresa nos plantea dos estrategias distintas para resolverlo:

- (a) Usar el algoritmo de A\*. Definimos el estado como la asignación de máquinas a canales. El estado inicial consiste en asignar las máquinas  $M_I$  y  $M_F$  a cada canal. Tenemos un operador que asigna una máquina a un canal solo si no se excede el valor  $P$  para ese canal. El coste es la velocidad media del canal imaginario que empieza en  $M_I$ , pasa por la máquina que estamos asignando y acaba en  $M_F$ . Como función heurística usamos la suma de las velocidades medias de todos los canales imaginarios que se obtendrían poniendo cada máquina que queda por conectar como único nodo intermedio entre  $M_I$  y  $M_F$ .
- (b) Usar un algoritmo de satisfacción de restricciones. El grafo de restricciones tendría como variables las máquinas, los dominios son los canales donde podemos asignarlas. Supondremos que no tenemos en cuenta las máquinas  $M_I$  y  $M_F$  ya que estarán asignadas a todos los canales. Como restricciones imponemos que un canal no esté asignado a más de  $P$  máquinas y que la velocidad media mínima del canal que incluye las máquinas asignadas sea mayor que  $V$ .

Comenta cada una de las soluciones que se proponen, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

3. El aumento del precio del cable de fibra óptica en el mercado negro ha disparado los casos de robo de cableado de fibra óptica a gran escala, tanto de los cables que proporcionan conexión de internet a poblaciones como los cables que se usan en las vías del AVE para transmitir comunicaciones y señales de control. Esto ha disparado la demanda de cable de fibra óptica de diferentes longitudes, según lo grande del tramo robado. Una empresa se quiere especializar en servir de forma rápida y eficiente estos pedidos en el sur de Europa. Para ello, en vez de fabricar cables de fibra óptica a medida, planea tener pre-fabricados  $R$  cables de una única longitud fija ( $l$ ) que se empacan y almacenan en  $R$  rulos, y luego se sirven los pedidos cortando el cable del rulo en tramos de cable de la longitud solicitada (ningún tramo de cable solicitado será de mayor que  $l$ ).

Para reducir la cantidad y longitud de los tramos cortos de cable que no le sirven a nadie, nos han pedido hacer un sistema que, dado un conjunto de  $T$  pares ( $id\_solicitud, longitud$ ) obtenido al juntar todos los pedidos recibidos en un día, nos diga como hemos de cortar los cables de rulo pre-fabricados en tramos de cable de manera que desperdiciemos la menor cantidad de material.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística,...). Para cada una de ellas se ha de comentar la solución que se propone, analizando si la técnica escogida es adecuada para este

problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

- (a) Se plantea aplicar Hill-climbing, usando como solución inicial asignar (en orden creciente de longitud) tramos de cable solicitados a cables de rulo; cuando la suma de las longitudes de los tramos asignados a un cable superan su longitud se pasa al siguiente cable de rulo. Como operadores se usan `mover_un_tramo` solicitado de un cable de rulo a otro y `quitar_un_rulo` de cable que no tiene tramos asignados. Como función heurística se usa la longitud de cable desecharido, que se calcula como la diferencia entre 1) la suma de las longitudes de los tramos solicitados y 2) la longitud total de los cables de rulo que forman parte de la solución (o lo que es lo mismo,  $l$  multiplicado por el número de rulos con tramos asignados).
  - (b) Se plantea usar algoritmos genéticos. Para la representación de una solución se decide asociar primero cada cable de rulo a un número de 1 a  $R$ . La idea es asignar a cada tramo solicitado el número que identifica el cable de rulo que tiene asignado. Cada solución se codifica como una cadena de  $T \times \log_2 R$  bits, teniendo para cada tramo el identificador de rulo en binario. Como mecanismo para generar la población inicial se escoge al azar, para cada tramo solicitado, un número en binario dentro de los valores  $[1, R]$  codificando el cable de rulo asignado al tramo. Como operadores genéticos se usan los operadores de cruce y mutación habituales. La función heurística será el número de rulos a cortar distintos que hay en la codificación de la solución.
  - (c) Se plantea usar un algoritmo de programación de restricciones. Las variables son los tramos de cable de los pedidos, los dominios son un número natural (de 1 a  $R$ ) que identifica el rulo del que se cortará el tramo de cable del pedido. Como restricciones se propone una restricción n-aria que abarca todas las variables que comprueba que, para cada rulo, la suma de longitudes de los tramos de cable asignados al rulo no exceda  $l$ , y otra restricción n-aria para comprobar que la suma de longitudes de cable desecharido (no asignado a ningún pedido) sea menor que una longitud  $D$ .
4. Con la introducción de la telefonía 5G es necesario hacer un uso más inteligente del limitado rango de frecuencias en el que operan los teléfonos. Para ello se necesita de un sistema dinámico que asigne a los  $T$  teléfonos que están llamando en cierto momento la frecuencia en la que tienen que operar y la antena a través de la que tienen que comunicarse. Tenemos un conjunto de  $A$  antenas de telefonía, cada antena tiene asignadas  $f$  frecuencias que puede utilizar. Dos teléfonos móviles pueden operar en la misma frecuencia si la distancia que los separa es mayor que  $d_{min}$ . Una antena no puede manejar más de  $t_{max}$  teléfonos o  $tf_{max}$  teléfonos en la misma frecuencia, ni tampoco teléfonos que estén a una distancia de más de  $da_{max}$  de la antena.

El objetivo del sistema es encontrar una asignación de todos los  $T$  teléfonos móviles a antenas y frecuencias que cumpla las condiciones anteriores y que minimice la distancia entre los teléfonos y la antena asignada (a mayor distancia, el teléfono móvil ha de aumentar la intensidad de su señal, exponiendo al usuario a más radiación e incrementando el consumo de batería).

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (estado o solución inicial, operadores, función heurística,...). Para cada una de ellas se ha de comentar los diferentes elementos de la propuesta, analizando si la técnica escogida es adecuada para este problema, si cada uno sus elementos son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

- (a) Se plantea solucionarlo mediante A\*. Consideramos que el estado permite representar una asignación total o parcial de antena y frecuencia a los teléfonos móviles. Buscamos la secuencia de asignaciones que da un valor a todos los teléfonos móviles. Como operador usamos asignar una antena y una frecuencia a un teléfono siempre que la asignación no provoque interferencia por su distancia con otro teléfono ya asignado a la misma frecuencia y no supere los límites de número de teléfonos por antena y máximo de teléfonos con la misma frecuencia de la antena. El coste del

operador es la distancia del teléfono a la antena asignada. La función heurística es la suma de las distancias de cada uno de los teléfonos por asignar a su antena más cercana.

- (b) Usar Hill-Climbing. La solución inicial se construye asignando a cada teléfono su antena más cercana y una de las frecuencias de esa antena al azar. Tenemos un operador que desconecta un teléfono de la antena y frecuencia asignadas y otro operador que conecta un teléfono a una cierta antena y frecuencia, comprobando que no provoque interferencia por su distancia con otro teléfono ya asignado a la misma frecuencia y no supere los límites de número de teléfonos por antena y máximo de teléfonos con la misma frecuencia de la antena. La función heurística será la suma de las distancias de cada teléfono con la antena asignada (para teléfonos sin antena asignada, el valor es 0).
  - (c) Se plantea resolverlo como un problema de satisfacción de restricciones. Consideramos que los teléfonos móviles son variables y la frecuencia y la antena que se les ha de asignar son los dominios. Tenemos una restricción global que establece que dos teléfonos no pueden tener la misma frecuencia si su distancia es menor que  $d_{min}$ . Tenemos otra restricción global que impide que asignemos la misma antena a más de  $t_{max}$  teléfonos. Antes de iniciar la solución del problema eliminamos del dominio de cada teléfono todas las antenas que están a una distancia de más de  $da_{max}$ .
5. La crisis militar en Ucrania ha provocado que miles de familias inmigrantes deban ser acogidas por los países de la Unión Europea. Como parte de esta, la Generalitat de Cataluña ya ha recibido una cantidad de familias de refugiados en sus comarcas, y deben tomarse medidas para asegurar su adaptación y bienestar. Partimos de que hay  $C$  comarcas en Cataluña, con  $I_c$  familias inmigrantes actualmente en cada comarca. Para asegurar su bienestar, el Govern de la Generalitat puede distribuir paquetes de ayuda en tamaños estandarizados de 100.000 o 500.000€, tantos como quiera por comarca. Por cada familia inmigrante, se establece un mínimo de  $A$  (ayuda) euros de presupuesto que debería obtener la comarca para asegurar su bienestar.

Para flexibilizar la situación, el Govern puede ayudar con la redistribución de las familias de inmigrantes, desplazando una cantidad fija ( $K$ ) de familias de su comarca actual a otra, pero este desplazamiento deberá ir acompañado de una indemnización a cada familia (con coste total de  $D$  euros), rebajando la cantidad de presupuesto en la comarca original por  $K * A$  y subiendo el de la de destino por la misma cantidad. Por otro lado, la Generalitat sabe que cada envío de paquete a una comarca tiene un coste burocrático en personal y tiempo, así que busca minimizar el total de paquetes que se envían. Además, cada comarca tiene un Consell Comarcal que puede o no estar alineado políticamente con el Govern de la Generalitat, y en el reparto de dinero el partido en el poder preferiría que el dinero total de ayudas que van a comarcas no alineadas sea menor, ya que esto le dará más poder en las siguientes elecciones.

Queremos pues, encontrar el reparto de ayudas y, si es necesario, de desplazamientos a hacer, con tal de minimizar la cantidad de dinero invertido, minimizar la cantidad total de paquetes de ayudas enviados y minimizar la cantidad de presupuesto que va a comarcas bajo partidos políticos no afines al del Govern. Al Govern le gustaría que el resultado fuera el óptimo si es posible; pero si fuera imposible aceptará sencillamente que los criterios sean optimizados. Se nos proponen las siguientes soluciones:

- (a) Usar el algoritmo de A\*. El estado inicial parte de ninguna ayuda asignada, y ningún desplazamiento de familias hecho. Como operadores tenemos dos: el operador `mover_refugiados` de una comarca a otra, con condición de aplicabilidad que haya  $\geq K$  refugiados en la comarca de origen y coste  $D$ ; operador `añadir_paquete` que añade un paquete de 100K o de 500K a una comarca, con coste  $coste(p) + \lambda + \epsilon * no\_afin(c) * coste(p)$ , donde  $coste(p)$  es el coste del paquete añadido (100K o 500K),  $\lambda$  y  $\epsilon$  son constantes dadas que tienen que ver con el coste burocrático y político (y  $no\_afin(c)$  vale 1 si la comarca no es afín y 0 si lo es); y el operador `quitar_paquete` que es el opuesto, con el mismo coste pero en negativo, y con condición de aplicabilidad que haya  $\geq 1$  ayuda del tipo escogido en la comarca. Como función heurística, usamos el número total de familias todavía no cubiertas por las ayudas; en otras palabras, la suma de, para cada comarca  $c$ :  $I_c - min(I_c, \lfloor \frac{suma_ayudas(c)}{A} \rfloor)$ , y donde entendemos que  $I_c$  ya tiene en cuenta los desplazamientos hechos con el operador `mover_refugiados`.

- (b) Usar el algoritmo de Hill Climbing. La representación de una solución incluye, para cada comarca, cuántas familias inmigrantes hemos desplazado, cuántas hay en cada comarca y cuántos paquetes de ayudas de cada tipo (100K, 500K) hay en cada comarca. Como solución inicial, partimos de una solución con 0 familias desplazadas, y en cada comarca hay el mínimo de ayudas de 100K necesarias para cubrir a todos los inmigrantes que hay inicialmente (es decir,  $\lceil \frac{I_c * A}{100K} \rceil$ ). Como operadores tenemos `mover_refugiados`, con condición de aplicabilidad que haya  $\geq K$  refugiados en la comarca de origen; `añadir_paquete` a una comarca, de 100K o de 500K; `quitar_paquete` de una comarca, con condición de aplicabilidad que haya  $\geq 1$  paquete de ese tipo en la comarca; `comprimir_paquete` que coge 5 paquetes de 100K y los sustituye por uno de 500K, condición de aplicabilidad que haya  $\geq 5$  paquetes de 100K en la comarca; `descomprimir_paquete` que hace lo opuesto, condición de aplicabilidad que haya 1 paquete de 500K en la comarca. Como función heurística usamos:  $D * \#desplazamientos + \sum_{c \in C} \lambda * \#\text{paquetes}(c) + \text{suma_ayudas}(c) + \epsilon * \text{no\_afin}(c) * \text{suma_ayudas}(c)$ , donde igual que en el apartado anterior,  $\lambda$  y  $\epsilon$  son constantes dadas que tienen que ver con el coste burocrático de cada ayuda y con el coste político de dar dinero a partidos opuestos, y  $\text{no\_afin}(c)$  vale 1 si la comarca es de un partido no afín al Govern y 0 si es afín.

Comenta cada una de las soluciones que se proponen, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

6. La huelga de funcionarios de la Administración de Justicia exigiendo mejoras salariales ha puesto presión en el gobierno, y nos piden que ayudemos con la reestructuración del sistema de justicia Español. El Estado quiere mejorar la eficiencia del sistema con un nuevo procedimiento de asignación de plazas de funcionarios a los juzgados. Cada funcionario  $f_i$  tiene un rol (juez, fiscal, abogado defensor, administrativo...), acepta un cierto sueldo mínimo ( $Pf_i$ ) y acepta una distancia máxima a recorrer entre su casa y su trabajo ( $Df_i$ ). Cada juzgado  $j_k$  dispone de  $x_r$  plazas nuevas por cada rol  $r$ . El ministerio paga cada plaza del rol  $r$  a un precio máximo determinado ( $P_r^k$ ) que es diferente para cada juzgado  $j_k$ , y dispone de una tabla de distancias entre las viviendas de los funcionarios y los juzgados.

Se quiere asignar funcionarios a juzgados de forma que se maximice el ahorro económico respecto a lo que inicialmente está dispuesto a pagar el ministerio, se minimicen las distancias que los funcionarios deberán recorrer a su trabajo, y se cubra el máximo número de plazas nuevas, priorizando este último factor respecto a los dos anteriores.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística,...). Para cada una de ellas se ha de comentar la solución que se propone, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

- (a) Queremos usar A\* definiendo el estado como la asignación de funcionarios a plazas nuevas. El estado inicial es la asignación vacía. Como operadores tenemos en primer lugar el de asignar un funcionario a una plaza (si su rol coincide y si el sueldo y la distancia son convenientes). El coste de este operador es  $Pf_i$  (el sueldo mínimo del médico asignado). También tenemos un operador de desasignar un profesional de una plaza, cuyo coste es  $-Pf_i$ . La función heurística  $h$  que se pretende usar es el sumatorio de los sueldos máximos determinados ( $P_r^k$ ) para las plazas que quedan por asignar.
- (b) Usar Hill-climbing a partir del estado en el que no se ha asignado ninguna plaza. Utilizar los operadores de asignar/desasignar funcionario a plaza si los roles son iguales y el sueldo que piden es menor que el precio que se quiere pagar ( $P_r^k$ ). Usar como función heurística la suma de sueldos mínimos que aceptan los funcionarios asignados, más la suma de distancias que deberán recorrer dichos funcionarios para ir a trabajar, todo ello dividido por el número de plazas asignadas.

- (c) Queremos usar satisfacció de restricciones. Como variables, en primer lugar, tenemos las plazas nuevas por cubrir, y sus dominios son el conjunto de funcionarios  $f_i$  que tienen el rol adecuado para dicha plaza. También tenemos un conjunto de variables para cada plaza cuyo dominio es el precio en euros al que se paga el sueldo de dicha plaza. También tenemos otro conjunto de variables para cada plaza con dominio la distancia entre la vivienda y el juzgado de la plaza. Por último, tenemos una variable 'objetivo' con valor numérico arbitrario (pero positivo). Como restricciones, tenemos una restricció n-aria que mira que todas las plazas tengan exactamente un funcionario asignado, un conjunto de restricciones binarias entre cada variable de plaza-funcionario y su correspondiente plaza-dinero (número de restricciones igual al número de plazas) que indica que la variable de plaza-dinero debe ser más grande que el sueldo mínimo del funcionario asignado a la variable plaza-funcionario. Tenemos una restricció similar para plaza-funcionario y plaza-distancia, donde la última toma como valor la distancia entre la vivienda del funcionario asignado y el juzgado de la plaza seleccionada. Tenemos una restricció N-aria que junta todas las variables de plaza-distancia, plaza-dinero y la variable objetivo haciendo que la última tome como valor la suma de todas las variables de distancia más la suma de todos los sueldos multiplicada por una constante positiva  $\lambda$ . Finalmente, marcamos al algoritmo que el problema es un problema de minimización.
7. Dada la sequia persistente de los últimos años y el aumento de la gravedad de los incendios, se nos ha planteado crear un sistema de IA que ayude a rediseñar la ubicación de los parques de bomberos en el territorio. Para solucionar el problema se propone dividir el territorio en una cuadrícula de  $N \times M$  posiciones, cada posición tiene asignado un factor de accesibilidad  $A$  que toma un valor entero entre 1 y 3 (1 es una zona de fácil acceso, 3 es una zona de difícil acceso).

Hemos de ubicar un total de  $P$  parques de bomberos, cada parque de bomberos puede tener entre uno y tres camiones. Tenemos un total de  $C$  camiones para repartir (obviamente  $C > P$ ).

Definimos el factor de seguridad de una posición como la suma para esa posición y todas las que la rodean del cociente entre el número de camiones de bomberos que hay en esa posición y el factor de accesibilidad de cada posición.

El objetivo es ubicar todos los parques de bomberos y repartir entre ellos todos los camiones maximizando el factor de seguridad global (la suma del factor de seguridad para todas las posiciones).

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. Justifica todas las respuestas.

- (a) Se plantea solucionarlo mediante A\*, recorriendo la cuadrícula de la esquina superior izquierda a la inferior derecha. El estado es la asignación que hemos hecho de parques de bomberos y camiones a las posiciones recorridas. Utilizamos como operadore poner un parque de bomberos, asignando uno, dos o tres camiones en el caso de ponerlo, el coste del operador es el número de camiones asignados más uno, o no ponerlo con coste uno. La función heurística es el número de posiciones que nos quedan por visitar y vale infinito si ya hemos asignado más de  $C$  camiones o  $P$  parques de bomberos.
- (b) Se plantea usar Hill Climbing. La solución inicial se construye en tres pasos: 1) se escoge de forma aleatoria la posición  $(x,y)$  de los  $P$  parques, asegurándose que es una posición válida dentro de la cuadrícula de  $N \times M$  posiciones, y que no se asignan dos o más parques a la misma posición; 2) se asigna 1 camión de bomberos a cada uno de los  $P$  parques, y cada uno de los camiones que quedan por colocar ( $C - P$  camiones) se asignan totalmente al azar entre los  $P$  parques. Hay dos operadores: uno que permite mover un parque  $p_i$  de su posición actual a otra posición válida (la posición de destino no tiene ya un parque asignado, y los camiones asignados a  $p_i$  se mueven con el parque a la posición de destino); y otro operador que permite mover un camión de bomberos de su parque actual a otro parque, asegurándose que en el parque al que se le asigna no se sobrepasan los 3 camiones asignados. La función heurística es la suma del factor de seguridad para todas las posiciones de la cuadrícula.

- (c) Se plantea resolverlo como un problema de satisfacción de restricciones. Consideramos que los  $P$  parques de bomberos son las variables, y el dominio de cada una de ellas está compuesto por pares  $\langle$ posición, camiones $\rangle$ , tantos como combinaciones posibles hay entre las  $N \times M$  posiciones de la cuadrícula y el número de camiones asignados al parque (entre 1 y 3). Tenemos un grafo de restricciones (binarias) que conecta cada parque con todos los demás para controlar (de dos en dos) que las posiciones asignadas a los parques son diferentes. Tenemos una restricción global que controla, en todo momento, que el número de camiones asignados a parques no exceda el número  $C$ , y otra restricción global que controla que el factor de seguridad global sea superior a un valor  $FSG_{min}$  (asumimos que durante la ejecución del algoritmo PSR existe una función auxiliar que es capaz de calcular el factor de seguridad de todas las posiciones teniendo en cuenta solo los parques de bomberos que tienen posición y camiones asignados, asumiendo que el resto de parques de bomberos y camiones no existen a la hora de hacer el cálculo).
8. El ayuntamiento de una pequeña ciudad quiere ofrecer a sus ciudadanos un servicio similar al Bicing barcelonés, pero a menor escala. Uno de los problemas a resolver es como mover las bicicletas de un punto de recogida a otro para intentar que en toda estación haya algunos puestos libres para dejar bicicletas, y que en toda estación haya alguna bicicleta disponible. El ayuntamiento ya tiene un sistema que decide que bicicletas hay que mover de una estación a otra, y ha subcontratado los servicios de un camión, que puede llevar un máximo de  $B$  bicicletas a la vez. Cada hora el camión ha de recoger y dejar bicicletas en diferentes estaciones del servicio distribuidas por la ciudad, siguiendo un listado del ayuntamiento de pares (estación\_origen, estación\_destino), y ha de hacerlo realizando el recorrido más corto posible, sin que se sobrepase en ningún momento el máximo de bicicletas que el camión puede cargar. Cada hora partimos de cierto punto de origen y volvemos a él, habiendo movido todas las bicicletas que el ayuntamiento ha solicitado de su estación origen a su estación destino. Para obtener el recorrido se dispone de un mapa de la ciudad que indica la distancia mínima entre cada par de estaciones por las que ha de pasar el camión.
- Puedes resolverlo mediante:
- (a) El algoritmo de A\*. El estado es el camino recorrido. Utilizamos como coste la longitud del camino actual. La función heurística vale infinito si el camión en el estado actual supera el número  $B$  de bicicletas que puede transportar y, en caso contrario, es la suma de las distancias de las estaciones por recorrer al origen. El operador aplicable es pasar de la estación actual a otra no visitada. Para evitar la necesidad de otro operador, el punto de origen y final del camión se modela como una estación más.
  - (b) Satisfacción de restricciones, donde las variables son todas las aristas del grafo de conexiones entre las estaciones a recorrer, éstas son variables booleanas e indican si pertenecen al camino a recorrer o no. Las restricciones son que debe haber exactamente dos aristas de un mismo vértice en la solución y que no se sobrepase el número  $B$  de bicicletas que el camión puede llevar a la vez en el recorrido formado por las aristas.
- 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.
9. Los organizadores de la pasarela Gaudí nos piden un sistema inteligente que ayude a planificar los desfiles de la próxima edición. El sistema ha de organizar diez desfiles para diez diseñadores, dos por día en un total de cinco días, sabiendo que algunos diseñadores se niegan a desfilar el mismo día que algún otro. Se han de asignar modelos a los desfiles ( $m$  por desfile) sin que la misma persona desfile más de tres veces en total durante la semana. Cada diseñador tiene también unas restricciones de altura y peso para sus modelos. Los organizadores de la pasarela han de pagar a los modelos y cada uno tiene su tarifa por participar en un desfile. Obviamente, los organizadores quieren pagar lo menos posible. Tras un análisis inicial del problema un compañero de nuestra empresa nos plantea dos estrategias distintas para resolverlo:

- (a) Usar el algoritmo de A\*. El estado es una asignación parcial de diseñadores y modelos a desfiles. Se definen dos operadores: `añadir_diseñador` a un desfile concreto, que comprueba que el diseñador no se haya añadido antes y que en caso de ser el segundo diseñador asignado del día, no haya incompatibilidad entre ambos diseñadores, el coste de este operador es siempre 1; y `añadir_modelo` a un desfile concreto, que comprueba que el desfile tenga ya diseñador asignado, que no haya más de  $m$  modelos en el desfile y que se cumplan las restricciones del diseñador, el coste de este operador es la tarifa del modelo asignado. Como función heurística usamos el número de diseñadores que falta incluir en los desfiles más el número de modelos que falta incluir en los desfiles multiplicado por la tarifa del modelo más caro.
- (b) Usar un algoritmo de satisfacción de restricciones. El grafo de restricciones tendría tres variables por cada uno de los modelos que representan los tres posibles desfiles que se le pueden asignar a cada modelo ( $desfile_1modelo_x$ ,  $desfile_2modelo_x$ ,  $desfile_3modelo_x$ ) y una variable por cada uno de los 10 diseñadores ( $diseñador_y$ ). El dominio de cada una de esas variables sería el identificador del desfile a asignar (un número entero entre el 1 y el 10, asumiendo que 1 y 2 son los desfiles del primer día, 3 y 4 son los desfiles del segundo día, y así sucesivamente). Como restricciones binarias habría: desigualdad entre todo par de variables de un mismo modelo  $x$  ( $desfile_zmodelo_x$  y  $desfile_wmodelo_x$ ) de modo que tengan asignado un desfile diferente; desigualdad entre todo par de variables  $diseñador_z$  y  $diseñador_w$  (cada diseñador ha de tener asignado un desfile diferente), y para cada modelo  $x$  que no cumple las restricciones de altura y peso de un diseñador  $y$ , establecer también una restricción binaria de desigualdad entre la variable  $diseñador_y$  y las tres variables del modelo  $x$  (el modelo  $x$  no puede coincidir en ninguno de sus desfiles asignados con el diseñador  $y$ ).

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.

10. Para hacer presión ante RENFE, los sindicatos de conductores de Cercanías en Barcelona quieren encontrar la forma de cumplir los requisitos mínimos que se les imponen sobre número de trenes, pero afectando al mayor número de usuarios, para que sus reclamaciones les den a ellos mayor poder en las negociaciones. Para ello han construido una tabla que, para cada hora y línea de cercanías, les dice el número de usuarios que viajan a esa hora por esa línea (es decir, la demanda), y el número mínimo de trenes que según RENFE han de pasar durante esa hora por esa línea para cumplir los servicios mínimos. Sabemos también que cada tren puede llevar como máximo  $P$  pasajeros. Hay además otras reglas que han de cumplir, y es que RENFE impone un número mínimo  $L_i$  total de trenes que han de circular por la linea  $i$  cada día, y un número mínimo  $H_h$  total de trenes que han de circular en una hora  $h$ , siendo estos números algo mayores que la suma de los trenes por lineas o por horas de los servicios mínimos antes mencionados.

Se nos plantean las siguientes alternativas:

- (a) Queremos utilizar satisfacción de restricciones donde tenemos una variable por cada línea de cercanías y cada hora, y los valores son el número de trenes asignados a cada línea y cada hora. Las restricciones son el número mínimo de trenes para cada línea y hora, el número mínimo de trenes que han de circular para cada línea durante el día, el número mínimo de trenes que han de circular para cada hora y que el número total de usuarios que se quedan sin tren sea menor que un cierto valor  $U$ .
- (b) Queremos utilizar búsqueda local, donde se genera una solución inicial colocando suficientes trenes para cubrir los servicios mínimos de cada línea y hora, y asignando aleatoriamente los trenes restantes entre todas las líneas. Los operadores de modificación de la solución consisten en mover un tren de una hora a otra en la misma línea, y mover un tren de una línea a otra. Queremos maximizar el número de usuarios que se verán afectados por la falta de trenes.

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.

11. Tras el proceso de reestructuración bancaria el potente Banco Nacional de Crédito (BNC) ha comprado la Caja de Ahorros Popular (CAP) con todas sus oficinas. El problema es que, tras la fusión de ambas entidades, hay un exceso de oficinas bancarias en una de las ciudades donde ambas entidades operaban antes de la fusión. Nos piden un sistema inteligente que les ayude a decidir qué oficinas cerrar. Tenemos un mapa de la ciudad con las  $O$  oficinas disponibles tras la fusión. Para cada oficina  $o_i$  podemos obtener el coste anual de su alquiler ( $\text{alquiler}(o_i)$ ). Tenemos también un listado de todos los  $C$  clientes que tenemos tras la fusión del BNC y la CAP. Para cada cliente  $c_j$  tenemos su dirección (geo-localizada en el mapa) y podemos obtener el importe total de los depósitos que tiene en la entidad ( $\text{depositos}(c_j)$ ). El banco le da mucha importancia a tener oficinas cerca de sus clientes. Por sus estudios de mercado han visto que sus mejores clientes suelen estar en un radio de 600 metros de sus oficinas. Nos dan una función `Oficina_oficina_cercana(Cliente c)` que dado un cliente  $c$  usa el mapa de la ciudad para devolvernos la oficina o más cercana, o devuelve null si no hay ninguna oficina a menos de 600 metros del cliente. También tienen una función `float beneficio_deposito(Cliente c)` que dado un cliente  $c$  hace una previsión de los beneficios que obtendrá el banco durante un año operando en los mercados financieros con el dinero de los depósitos de ese cliente. A partir de estas funciones definen dos más: la primera es `float estim_benef_depositos(Oficina o)` que calcula la suma de beneficios obtenidos por los depósitos de todos los clientes cuya oficina más cercana es  $o$ ; la segunda es `LClientes_clientes_sin_oficina()` que nos devuelve la lista de clientes que no tienen ninguna oficina a menos de 600 metros de donde viven. Todas estas funciones se pueden invocar varias veces durante la ejecución y nos darán diferentes resultados dependiendo de las oficinas que se quieran cerrar y de cuál quede más cerca de cada cliente en cada momento.

El objetivo es reducir el número de oficinas que se mantienen abiertas tras la fusión de ambas entidades, minimizando el coste total anual en alquileres de las oficinas que se mantienen abiertas y maximizando el beneficio total obtenido a partir de los depósitos de los clientes que tienen una oficina a menos de 600 metros de su vivienda. La idea es cerrar oficinas que no salen a cuenta mantener porque no tienen suficientes clientes para cubrir gastos con los beneficios obtenidos, o porque hay otra oficina en la cercanía que puede atender a esos clientes. Aunque no gusta perder clientes, es posible dejar algunos clientes sin oficina cercana si mantener dicha oficina genera más costes que beneficios.

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (estado o solución inicial, operadores, función heurística,...). Para cada una de ellas se ha de comentar los diferentes elementos de la propuesta, analizando si la técnica escogida es adecuada para este problema, si cada uno sus elementos son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

- (a) Se plantea solucionarlo mediante A\*. Para ello se parte del estado en el que no hay ninguna oficina en la ciudad. Tenemos dos operadores: colocar oficina, que añade una oficina  $o_i \in O$ , su coste es  $\text{estim_benef_depositos}(o_i) - \text{alquiler}(o_i)$ ; eliminar oficina, que elimina una oficina  $o_i$  que hubiera sido colocada con el operador anterior, su coste es  $-(\text{estim_benef_depositos}(o_i) - \text{alquiler}(o_i))$ . Como heurístico  $h$  se usa:

$$h_2 = \sum_{\forall c_j \in CN} \text{beneficio_deposito}(c_j)$$

siendo  $CN$  el conjunto de clientes que quedan por servir (los que no tienen ninguna oficina cercana), que se obtiene con la función `clientes_sin_oficina()`.

- (b) Se plantea solucionarlo mediante Hill-Climbing. Como solución inicial se parte del conjunto completo de todas las  $O$  oficinas disponibles y de los  $C$  clientes. Como operador único se plantea el eliminar una oficina de la solución. Como función heurística se usa

$$h_1 = \sum_{\forall o_i \in OS} \text{estim_benef_depositos}(o_i) - \text{alquiler}(o_i)$$

siendo  $OS$  el conjunto de oficinas supervivientes (no eliminadas).

- (c) Se plantea resolverlo mediante algoritmos genéticos. Para representar el problema utilizamos una tira de  $O$  bits, donde un bit a 1 indica que esa oficina se va a mantener abierta y un bit a 0 indica que esa oficina se va a cerrar. Como población inicial generamos aleatoriamente  $n$  individuos donde en cada uno hay  $O/2$  bits a 1. Como operadores usamos un operador de cruce en un punto y un operador de mutación que cambia el valor de un bit de la cadena (de 0 a 1 o de 1 a 0). La función heurística es:

$$h_3 = \sum_{\forall o_i \in OS} \frac{\text{estim\_benef\_depositos}(o_i)}{\text{alquiler}(o_i)}$$

siendo  $OS$  el conjunto de oficinas supervivientes (no eliminadas), es decir, las que tienen el bit a 1.

12. El mago Rincewind debe componer un hechizo a partir de conocimiento arcano que se encuentra en diversas torres esparcidas por el mundo, cada una con una parte única y indispensable. Hay un total de  $T$  torres, y desde cada torre se puede viajar a cualquier otra torre, pero la distancia puede variar y a Rincewind no le gusta andar sin sentido, así que quiere hacer el recorrido con menor distancia posible.

Sin embargo, el problema no acaba aquí. **Una proporción bastante alta** de estas torres se encuentran cerrada bajo sellos arcanos que requieren de llaves (en forma de hechizos). Estas llaves se encuentran en algunas de las otras torres que Rincewind tiene que visitar. Por ejemplo, para visitar(y acceder a) la torre A Rincewind requiere de haber visitado la torre B o la torre C (ambas contienen el hechizo para abrir la torre A). Generalmente, el hechizo de cada torre se puede encontrar en una o dos torres, no en muchas más. En consecuencia, el recorrido de Rincewind debe tener en cuenta un conjunto de preordenes entre las torres. El sistema está montado de forma que todas las torres son accesibles. Es decir, no puede darse que haya un ciclo entre los preordenes como que el hechizo que abre la torre A está solo en la torre B y el que abre la torre B, solo en A (este problema no tendría solución).

Rincewind dispone de una serie de propuestas para organizar la ruta, que puede programar para que Hex (el superordenador de la universidad) saque como salida el orden de visita a las torres, minimizando la distancia.

- (a) Usar el algoritmo de A\*. Definimos el estado como la asignación de torres a la ruta. El estado inicial es la ruta vacía. Tenemos un operador `visitar_torre` que asigna la siguiente torre a la ruta, el coste es la distancia entre la torre actual y la siguiente escogida. Como función heurística usamos  $h(n) = M * P + k * L$ , donde  $M$  es la distancia media entre dos torres,  $P$  es el número de torres a las que todavía no hemos ido (*pending*),  $k$  es un factor de ponderación y  $L$  es el número de torres todavía selladas (*locked*, es decir, el número de llaves por recoger).
- (b) Usar el algoritmo de Hill Climbing. La solución inicial se construye añadiendo todas las torres en orden alfabético. Como operador de búsqueda usamos `swap(torre1, torre2)`, que comprueba que el intercambio de torres en la ruta cumpla los pre-ordenes impuestos por los sellos. La función heurística es la suma de las distancias entre torres consecutivas en la solución.
- (c) Usar un algoritmo de satisfacción de restricciones. El grafo de restricciones tendría como variables las torres, los dominios son un número natural (de 1 a  $T$ ) indicando el orden de cada torre en la ruta. Como restricciones se propone una restricción n-aria tal que se conserven los preordenes establecidos por los sellos arcanos, y otra restricción n-aria para comprobar que la distancia de la ruta sea menor que una distancia  $D$  (que iremos disminuyendo y re-ejecutando a medida que encontramos soluciones hasta encontrar la óptima).

Comenta cada una de las soluciones que se proponen, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

13. Un problema logístico en el proceso de gestión y acogida de los desplazados por la guerra es asegurar transporte para los voluntarios que no disponen de vehículo propio y que han de desplazarse por zonas

del territorio con pocas opciones de transporte público. Por ello los alcaldes de los distritos de Briceni, Dondușeni, Edineț y Ocnița (en la punta norte de Moldavia) han alquilado de forma conjunta un autocar con 40 plazas que hará un recorrido cada hora, y nos han pedido crear un programa que, dada la información sobre el punto de recogida de cada voluntario y su punto de destino (ambos dentro de la zona de los cuatro distritos) calcule el recorrido más corto posible que debe hacer el autocar para recoger y dejar a todos los voluntarios, sin que se sobrepase en ningún momento el número de plazas disponibles en el autocar. Cada hora se calculará un recorrido únicamente con los voluntarios que se han de desplazar durante esa hora.

El autocar parte desde cierto punto de origen y ha de volver a él al final del recorrido, habiendo dejado todos los voluntarios en sus puntos de destino. En ningún punto de recogida habrá más de 40 voluntarios por recoger. Para obtener el recorrido se dispone de un mapa de los distritos que indica la longitud del camino mínimo entre cada dos puntos por los que ha de pasar el autocar.

- (a) Queremos utilizar A\*. Usaremos como coste la longitud del camino. La función heurística vale infinito si el autocar supera el número máximo de pasajeros y en caso contrario, es la suma de las distancias de los puntos por recorrer al origen. El operador aplicable es pasar del punto actual a otro no visitado (se supone que en cada punto visitado se recoge o deja a todos los voluntarios que toca, no se puede recoger solo a una parte de los voluntarios y volver más tarde a por el resto).
- (b) Aplicar un algoritmo de satisfacción de restricciones. Las variables son todas las aristas del grafo de conexiones entre los puntos a recorrer, éstas son booleanas e indican si pertenecen al camino a recorrer o no. Las restricciones son que debe haber exactamente dos aristas de un mismo vértice en la solución y que no se sobrepase el número máximo de pasajeros del autocar en el recorrido.

Comenta cada una de las soluciones que se proponen, analizando si la técnica escogida es adecuada para este problema, si cada uno de los elementos de la solución son correctos o no (cada uno por separado y en conjunción los unos con los otros). Incluye un análisis de los costes algorítmicos y/o factores de ramificación allá donde sea necesario. Justifica tu respuesta.

14. Tras las numerosas críticas y movilizaciones de los taxistas por el mecanismo de turnos que ha implantado recientemente el *Ajuntament de Barcelona* para el sector, un grupo de trabajo formado por representantes de los taxistas y del *Ajuntament* nos ha pedido crear un sistema que adapte el número de taxis circulando por el área metropolitana a la demanda estimada y al nivel de contaminación diaria (medida en gr. de CO<sub>2</sub> emitidos a la atmósfera).

Según nos cuentan, en estos momentos hay un número  $T$  de taxis registrados en el área metropolitana. Nos han contado que en vez de controlar el número de pasajeros que lleva cada taxi, el sistema controlará el número de servicios (un servicio es un viaje del taxi que recoge a un cierto número de pasajeros en un lugar de origen y los deja en un lugar de destino). Para simplificar el problema consideraremos que cada servicio dura unos 15 minutos de media. Se establece que el sistema asignará a cada taxi como mínimo  $NS$  servicios a la semana. Para cada taxi  $t$  disponemos del valor  $CO_t$  que mide el grado de contaminación que produce (medido en gr. de CO<sub>2</sub> emitidos por minuto). El *Ajuntament* nos dice que en ningún momento los taxis en servicio han de superar la cota máxima  $MAX\_CO\_H$  de gramos de CO<sub>2</sub> generados en una hora, y que en un día la cantidad de gramos de CO<sub>2</sub> emitidos no puede superar la cota  $MAX\_CO\_D$ . Con el número  $T$  de taxis disponibles es fácil superar ambas cotas. zona, tenemos una tabla semanal en la que, para cada zona  $z$ , día  $d$  y hora  $h$  se tiene una estimación del número de servicios ( $S_{z,d,h}$ ) que se suelen originar en esa zona, para cada hora del día. Para modelar la demanda esperada de taxis tenemos una tabla semanal en la que, para cada día  $d$  y hora  $h$  se tiene una estimación del número de servicios ( $S_{d,h}$ ) que se suelen realizar.

El objetivo es decidir de forma dinámica los taxis que han de circular por el área metropolitana, de forma que se minimicen los niveles de contaminación diaria (nunca se pueden superar los límites) y se maximice el número de servicios realizados (pero podemos dejar usuarios sin servir, si añadir servicios implica superar los límites de contaminación). Se nos plantean las siguientes alternativas:

- (a) Queremos utilizar A\*. Definimos el estado como la asignación de taxis a horas del día. El estado inicial consiste en asignar de forma ordenada a cada taxi un número mínimo de servicios  $NS$  distribuidos de forma aleatoria en las horas de la semana. Tenemos un operador que asigna un nuevo servicio a un taxi  $t$  en un dia  $d$  y hora  $h$  determinados, el coste serán los gramos  $CO_t$  emitidos en los 15 minutos del servicio. Como función heurística usamos la suma del numero de servicios estimados en la tabla de demanda y que aun no hemos servido, o infinito si no se supera la cota  $S_{d,h}$ .
- (b) Queremos utilizar Hill Climbing, donde se genera una solución inicial asignando al azar suficientes taxis a servicios hasta llegar a servir toda la demanda estimada en cada hora. Los operadores de modificación de la solución consisten en intercambiar dos taxis entre servicios que pertenezcan a horas diferentes, y eliminar un servicio asignado a un taxi (siempre que no quedemos por debajo del límite  $NS$  de servicios mínimos asignados a ese taxi). Como función de evaluación usaremos el sumatorio de los gramos de  $CO_2$  emitidos por todos los taxis asignados a servicios en la solución actual.

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.

15. El mago Rincewind se encuentra en unas ruinas antiguas con un mecanismo similar a un circuito. El espacio del mecanismo está compuesto por  $C$  columnas verticales, cada una de  $H$  altura máxima. Hay un conjunto de piezas diferentes en la sala para componer el circuito dentro de este espacio. Cada pieza puede ocupar una única columna, y tiene una altura  $h_p$ . Una pieza se puede colocar en el suelo de la columna, o encima de otra pieza ya colocada, pero nunca flotando en el aire. Además, las piezas tienen por cada lado una configuración de *thaumaje* o flujo mágico. Por ejemplo, una pieza de altura 3 puede tener *thaumaje* (de abajo a arriba) *azul-verde-amarillo* por un lado, y *azul-azul-azul* por el otro lado. El *thaumaje* es importante, porque dos piezas en columnas colindantes que comparten una altura con el mismo color están conectadas.

Además, hay dos puertos, uno en cada lado de la terminación del circuito (izquierda y derecha), a alturas diferentes  $H_i$  y  $H_d$ , con *thaumaje* diferente. El objetivo del mecanismo es conectar ambos puertos mediante las piezas que hay en la sala, para lo cual debe existir un camino de piezas conectadas de forma que ambos extremos estén también conectados a los puertos (ej. que la pieza más a la izquierda del camino tenga el *thaumaje* correcto a la altura  $H_i$ ).

Para resolver el mecanismo es necesario que no haya piezas flotando, que los puertos estén conectados como se ha descrito, que no haya piezas superimpuestas y que no haya piezas que superen la altura máxima del mecanismo  $H$ . Además, cada región colindante entre dos piezas que tenga *thaumaje* diferente implica una pérdida de flujo importante, y se debe minimizar. Por último, Rincewind es un vago, y quiere minimizar el número de piezas pesadas que le toca poner en el mecanismo. El peso de una pieza está descrito como  $W_p$ .

En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone respecto a si es correcta, es eficiente, o es mejor o peor respecto a otras alternativas posibles para el mismo algoritmo. Justifica tu respuesta.

- (a) Se plantea solucionarlo mediante Hill-Climbing. Empezamos desde la solución vacía. Como operadores, tenemos un operador `colocar(P,C)` que coloca la pieza  $P$  en la columna  $C$  encima de las que ya haya y `swap(P1,P2)` que intercambia dos piezas colocadas en el mecanismo. Como heurística se propone el número total de thaumajes incompatibles en la solución más el número de piezas conectadas al puerto izquierdo y derecho al cuadrado (por separado), más el peso total de las piezas asignadas a la solución actual:

$$\sum_{c \in \{1..C-1\}} \sum_{h \in H} th\_incompat(c, c+1, h) + conectadas\_izq()^2 + conectadas\_der()^2 + \sum_{p \in P} W_p * colocada(P) \quad (6.1)$$

- (b) Se plantea resolverlo mediante algoritmos genéticos. La representación consiste en una ristra de  $P * (\lceil \log_2 C \rceil + N)$  bits. Cada uno de los  $P$  bloques representa en binario la id de la columna donde está el pieza, y un número en binario que 'desempata' piezas de la misma columna y marca el orden en alturas (por ejemplo: "010(C)-11(N) | 010(C)-00(N)" implica que ambas piezas van a la columna 3 (010) y que la pieza 1 está encima de la pieza 2 (porque 3 es más grande que 0). En caso de empate, se ordenan en orden de id de pieza. Como heurística usamos la siguiente:

$$\text{conectadas\_a\_puerto}() - \lambda * \sum_{p \in P} W_p * \text{colocada}(P) - \gamma * \sum_{c \in \{1..C-1\}} \sum_{h \in H} \text{th\_incompat}(c, c+1, h) \quad (6.2)$$

- (c) Se plantea resolverlo mediante CSP. Tenemos  $P$  variables cuyos dominios son las  $C$  columnas más la no-asignación. Tenemos también una variable  $W$  de peso total, y una variable  $I$  que cuenta las incompatibilidades de flujo. Por último, tenemos una variable  $O$  que suma ambas, y se pide minimizar. Como restricciones, tenemos  $P$  restricciones de preprocesado que impide poner piezas que se sobrepasen de la altura  $H$ , una n-aria entre  $P$  y la variable  $I$  que hace que  $I$  valga el número de incompatibilidades en la asignación actual, una restricción similar para  $W$  y una restricción ternaria entre  $W$ ,  $I$  y  $O$  que hace que la última valga la suma de las dos anteriores. Por último, tenemos una restricción N-aria entre las  $P$  variables que comprueba que ambos puertos están conectados a través de las piezas colocadas.
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 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.
- (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.
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 las siguientes alternativas para minimizar el número total de pasos de ejecución del meta-servicio:

- (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:

- 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
- 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.

- (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:

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

18. El Servei Català de la Salut (CatSalut) quiere mejorar el sistema de asignación de plazas de médicos a centros hospitalarios catalanes. Cada médico  $m_i$  tiene una especialidad (medicina general, pediatría, traumatología, dermatología, odontología, cardiología, oncología, ...), acepta un sueldo mínimo determinado ( $Sm_i$ ) y acepta una distancia máxima a recorrer entre su casa y su trabajo ( $Dm_i$ ). Cada centro  $c_j$  dispone de  $x_e$  plazas nuevas por cada especialidad  $e$ . Para cada plaza de especialidad  $e$  el CatSalut determina un sueldo máximo ( $S_e^j$ ) que es diferente para cada centro  $c_j$ , y dispone de una función `float distancia(Medico m, Centro c)` que devuelve la distancia que hay (en km) entre la vivienda de un médico  $m$  y un centro  $c$  dados.

Se quiere asignar médicos a centros de manera que se minimicen las distancias que los médicos deberán recorrer cada día (para mejorar su rendimiento), se maximice el ahorro económico respecto a lo que inicialmente está dispuesto a pagar CatSalut por plaza ( $S_e^j - Sm_i$ ) y cubra el máximo número de plazas nuevas, priorizando este último criterio respecto a los dos anteriores. Supondremos que el número de médicos que solicitan plaza es mucho mayor que el número de plazas, pero las restricciones de especialidad, sueldo y distancia pueden dejar plazas nuevas sin asignar.

Se nos plantean las siguientes formas de solucionar automáticamente este problema:

- (a) Queremos usar A\* definiendo el estado como la asignación de médicos a plazas nuevas. El estado inicial es la asignación vacía. Como operadores tenemos el de asignar un médico a una plaza (si las especialidades coinciden y si el sueldo y la distancia son convenientes), el coste de este operador es  $Sm_i$  (el sueldo mínimo del médico asignado), y desasignar un médico de una plaza, cuyo coste es  $-Sm_i$ . La función heurística  $h$  que se pretende usar es el sumatorio de los sueldos máximos determinados ( $S_e^j$ ) para las plazas que quedan por asignar.
- (b) Queremos usar satisfacción de restricciones donde las variables son las plazas nuevas por cubrir, y sus dominios son el conjunto de médicos solicitantes  $m_i$  que tienen la especialidad adecuada para

cada una de las plazas. Tenemos un grafo de restricciones que conecta las plazas de manera que 2 plazas no puedan tener el mismo médico asignado. También se añaden como restricciones que el sueldo máximo determinado para la plaza sea mayor que el sueldo mínimo del médico asignado, y que la distancia a recorrer por el médico asignado sea menor que su distancia máxima. Tenemos además una restricción global que impone que el ahorro económico sea mayor que un valor  $minA$  dado.

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.

19. Una empresa de productos orgánicos nos pide un sistema inteligente para distribuir un conjunto de plantas en un cultivo. Para organizar mejor el cultivo se ha dividido en una cuadrícula de  $N \times M$  posiciones. Las plantas que quieren colocar son de  $T$  diferentes tipos y tienen  $k_t$  plantas de cada tipo. Cada planta tiene unas necesidades específicas de agua ( $a_t$ ) y nutrientes ( $n_t$ ) diarias.

Cada posición de la cuadrícula puede albergar un máximo de  $P$  plantas de cualquier tipo. El sistema de riego permite aportar  $L$  litros de agua diarios a cada una de las  $M$  columnas de la cuadrícula y de cada posición las plantas pueden consumir hasta  $G$  gramos de nutrientes al día.

El sistema inteligente ha de colocar todas las plantas, con las restricciones de que el consumo de agua y nutrientes no superen las cantidades diarias y que, para agotar de manera uniforme los nutrientes de las posiciones, la diferencia en consumo de nutrientes entre una posición y cualquiera de sus adyacentes no supere cierto valor  $C$ .

Tras un análisis inicial del problema un compañero de nuestra empresa nos plantea dos estrategias distintas para resolverlo:

- (a) Usar el algoritmo de A\*. El estado es la asignación de plantas a las casillas de la cuadrícula. Usamos como operador el añadir una planta de un tipo a una posición de la cuadrícula siempre que no se supere el número  $P$  de plantas en la posición y no se superen los consumos de agua y nutrientes indicados en el enunciado. El coste del operador es la suma de necesidades de agua y nutrientes de la planta colocada. La función heurística es la suma de agua y nutrientes de las plantas que quedan por colocar o infinito si en la solución actual hay alguna posición en la que la diferencia de consumo entre una posición y cualquiera de sus adyacentes es mayor que  $C$ .
- (b) Usar un algoritmo de satisfacción de restricciones. El grafo de restricciones tendría como variables las coordenadas de la cuadrícula del cultivo y los valores el identificador de cada planta a colocar (evidentemente una variable tendrá un conjunto de valores). Las restricciones aparecerían entre las posiciones de cada columna, de manera que las necesidades de agua totales no superen los  $L$  litros de agua que se aportan diariamente, habrá una restricción por posición que no permita que el consumo de nutrientes supere el límite  $G$  diario y restricciones entre una posición y sus adyacentes de manera que la diferencia de consumo de nutrientes no sea mayor que  $C$ .

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.

20. Una empresa pesquera y conservera envía sus  $N$  barcos  $\{b_i\}$  con capacidades para  $C(b_i)$  pescadores a sus  $D$  destinos internacionales  $\{d_i\}$  ( $N > D$ ) para 180 días de pesca. Algunos de sus  $M$  pescadores  $\{p_i\}$  ( $M > \sum C(b_i)$ ) son asignados a los barcos y son enviados a los destinos en el barco asignado. El coste por marinero del viaje en el barco  $b_i$  a un destino  $d_j$  es  $K_b(d_j, b_i)$ . Cada pescador  $p_i$  proporciona a la empresa una ganancia de  $g(p_i)$  euros/día en media y cuesta a la empresa  $k(p_i)$  euros/día en media. La empresa requiere una planificación que cubra todos sus destinos sin pérdidas.

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

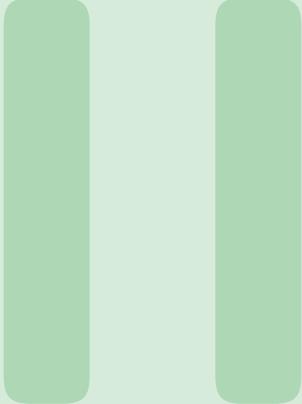
- (a) Aplicar búsqueda local. La solución inicial consiste en: 1) asignar secuencialmente  $N_{divD}$  barcos a cada destino, repartiendo los  $N_{modD}$  restantes equitativamente entre los primeros  $N_{modD}$  destinos; 2) asignar aleatoriamente  $C(b_i)$  pescadores a cada barco  $b_i$ . Los operadores son: cambiar el destino de un barco (y el de todos sus pescadores) e intercambiar dos pescadores entre dos barcos de destinos diferentes. Como función heurística usamos:

$$h'(n) = \sum_{i=1}^D \sum_{\forall b_j \in Asig(d_i)} K_b(d_i, b_j) + 180 \times \sum_{\forall p_i \in O(b_j)} (k(p_i) - g(p_i))$$

donde  $O(b_j)$  es la asignación de marineros del barco  $b_j$  ( $|O(b_j)| \leq C(b_j)$ ) y  $Asig(d_i)$  son los barcos asignados al destino  $d_i$ .

- (b) Aplicar satisfacción de restricciones. Las variables son todos los posibles pares  $\langle p_i, d_j \rangle$  (pescador, destino). Todas las variables tienen como dominio el conjunto  $\{b_i, undef\}$ . El primer valor denota que el barco  $b_i$  se asigna al destino y que el pescador llegará en él. El segundo valor denota que el pescador no viaja al destino. Las restricciones son:

$$R1 : \forall_i |O(b_i)| \leq C(b_i) \quad R2 : \sum_{j=1}^N \sum_{p_i \in O(b_j)} (k(p_i) - g(p_i)) \geq 0$$



# Planificació





## 7. Planificación

1. Traducir el siguiente modelo del problema del mundo de los bloques de una codificación STRIPS a una codificación en PDDL.

**Operadores:**

OP: pick-up(x):

PRE: clear(x), ontable(x), handempty().  
DEL: clear(x), ontable(x), handempty().  
ADD: holding(x).

OP: put-down(x):

PRE: holding(x).  
DEL: holding(x).  
ADD: clear(x), handempty(), ontable(x).

OP: stack(x,y):

PRE: holding(x), clear(y).  
DEL: holding(x), clear(y).  
ADD: clear(x), handempty(), on(x,y).

OP: unstack(x,y):

PRE: on(x,y), clear(x), handempty().  
DEL: on(x,y), clear(x), handempty().  
ADD: holding(x), clear(y).

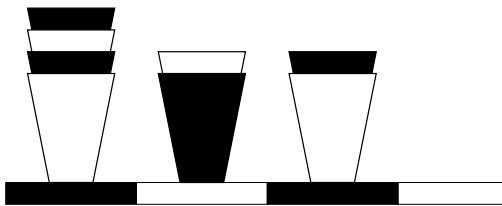
**Problema:**

```
clear(A), clear(B), clear(E), clear(C), clear(D), ontable(F), ontable(B),  
ontable(E), ontable(C), ontable(D), on(A,F), handempty().
```

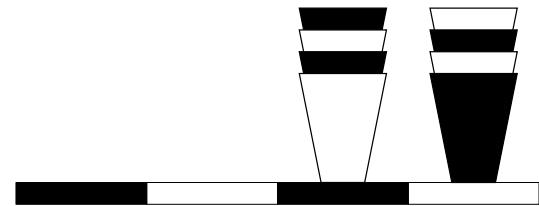
`on(E,F),on(F,C),on(C,B),on(B,A),on(A,D).`

2. Tenemos un vector de tres posiciones. Queremos crear un modelo de dominio que permita darle a un planificador una configuración inicial de valores del vector (ej: |a|b|c|) y una configuración final de esos valores en el vector (ej: |c|b|a|) y que un planificador estilo Fast Forward encuentre los movimientos necesarios a realizar.
3. Un equipo de investigadores en robótica necesita dotar a su robot de la capacidad de planificar acciones para poder probar su habilidad de reconocimiento de objetos y colores. Para empezar las pruebas quieren hacer un planificador simple que le diga al robot cómo apilar y desapilar vasos en una mesa. Los vasos pueden ser de dos colores: blanco y negro. En la mesa se han pintado cuatro casillas, dos blancas y dos negras. Las restricciones que se imponen son las siguientes:

- El robot solo puede usar uno de sus brazos, y coger un vaso a la vez.
  - Un vaso negro solo se puede poner encima de un vaso blanco o una casilla blanca.
  - Un vaso blanco solo se puede poner encima de un vaso negro o una casilla negra.
- (a) Describir el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Dad una explicación razonada de los elementos que habéis escogido. Tened en cuenta que el modelo del dominio ha de funcionar para tamaños de problema grandes (como el descrito en el apartado c)
  - (b) Describir el problema siguiente usando PDDL. Dad una breve explicación de cómo modeláis el problema.



Estado inicial



Estado final

- (c) Ahora supongamos que hemos pintado en la mesa un tablero de  $24 \times 24$  casillas, la mitad blancas y la mitad negras, y que tenemos 600 vasos blancos y 600 vasos negros. ¿Cuál sería el factor de ramificación de vuestros operadores? ¿Se os ocurre una forma de mejorar el modelo que habéis hecho? Razonad vuestra respuesta (no hace falta que cambiéis vuestro modelo, sólo que indiquéis qué cambios habría que hacer.).
4. El Departament de Salut está buscando un sistema inteligente que ayude a gestionar el envío inmediato de material bio-sanitario dentro de una área sanitaria. Para ello ha decidido crear una pequeña flota de vehículos eléctricos compuesta por mini-vans y motos eléctricas. A través de un servicio web diferentes entidades (hospitales, laboratorios de análisis, residencias de ancianos) podrán solicitar el envío de paquetes con material bio-sanitario: guantes, máscaras, equipos de protección individual (EPIs), gel hidroalcohólico, jabón desinfectante, toallas secantes de papel, pruebas PCR, muestras de sangre y material de desecho a incinerar.

Un planificador recibe las peticiones recibidas en los últimos 20 minutos (cada una de ellas incluye el material requerido, el lugar donde se ha de recoger el material y el lugar de destino en el que se ha de entregar el material) y crea un plan para servir las peticiones con los vehículos disponibles. Los vehículos salen de un parking municipal y después de hacer todos los transportes requeridos han de volver al parking para recargar sus baterías. Todos los paquetes son de una medida estandar, y los vehículos han sido preparados con contenedores especiales para transportar paquetes de esa medida.

En el caso de las mini-vans disponen de entre 2 y 6 contenedores especiales refrigerados, mientras que las motos eléctricas disponen de un único contenedor especial que no está refrigerado. Hay materiales (como los PCRs y las muestras de Sangre) que solo se pueden transportar dentro de un contenedor refrigerado. Pero para aprovechar los viajes se permite a las mini-vans que lleven también materiales que no precisen refrigeración en sus contenedores refrigerados. Cada vehículo solo puede llevar un paquete de material por contenedor, y no puede coger más paquetes si tiene todos sus contenedores llenos.

El plan generado ha de mostrar claramente como se mueve cada vehículo de un lugar a otro, y lo que van recogiendo y entregando en cada lugar.

- (a) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Explica el significado de cada predicado y sus parámetros (por ejemplo:  $(encima ?X ?Y) = el\ elemento\ ?X\ está\ encima\ del\ elemento\ ?Y$ ). Explica también si añades predicados para filtrar posibles asignaciones o para guiar mejor al planificador hacia el objetivo. Ten en cuenta que el modelo del dominio ha de poderse extender a más o menos paquetes, lugares de recogida y entrega y vehículos (con más o menos contenedores de transporte).
- (b) Nos han proporcionado una tabla como ejemplo de las peticiones que el sistema recibe cada 20 minutos, con una serie de paquetes de material (donde el \* indica que ha de transportarse refrigerado) que hay que transportar entre diferentes lugares: dos almacenes, tres hospitales, tres residencias de ancianos, un laboratorio de análisis y una incineradora. También nos dice que al inicio de ese intervalo de 20 minutos tenemos solo dos vehículos eléctricos disponibles: la *Moto1* (con capacidad para 1 paquete, sin refrigeración) y la *Van2* (con capacidad para transportar hasta 3 paquetes, tanto que requieran refrigerado como no). Ambos vehículos se encuentran aparcados en un parking municipal y, después de transportar todas las peticiones, han de volver al parking para poder recargar sus baterías eléctricas. Se nos pide que el planificador genere un plan en el que los vehículos disponibles sirvan todos los pedidos, transportando cada paquete de material del lugar de origen al de destino.

id_pedido	origen	destino	paquete (* = refrigerado)
p1	Almacen1	ResidenciaSants	Guantes
p2	Almacen1	HospitalSantPau	EPIs
p3	Almacen1	ResidenciaCollblanc	Mascaras
p4	Almacen2	HospitalVallHebron	JabonesDesinfectantes
p5	HospitalClinic	Incineradora	Deshechos
p6	Almacen1	HospitalClinic	Mascaras
p7	Almacen1	ResidenciaTresTorres	EPIs
p8	Almacen2	ResidenciaTresTorres	GelesHidroalcohólicos
p9	Almacen1	ResidenciaSants	ToallasPapel
p10	Almacen2	HospitalVallHebron	PCRs*
p11	HospitalClinic	Laboratorio	PCRs*
p12	HospitalSantPau	Laboratorio	MuestrasSangre*
p13	Almacen1	HospitalSantPau	Mascaras
p14	Almacen2	HospitalVallHebron	EPIs
p15	Almacen2	ResidenciaCollblanc	GelesHidroalcohólicos
p16	HospitalVallHebron	Laboratorio	PCRs*

Describe este problema usando PDDL. Da una breve explicación de cómo modelas el problema.

5. Una empresa de logística tiene que gestionar la entrega de paquetes voluminosos desde sus almacenes a una serie de domicilios en diversas localidades de una región. Para ello, disponen de un conjunto de vehículos de diversa capacidad, que pueden desplazarse libremente dentro de una localidad, o a través del conjunto de carreteras.

La empresa busca automatizar la organización de sus almacenes y vehículos para la entrega de todos los paquetes de una jornada. Previo al reparto local, la empresa puede enviar cualquier paquete desde

su centralita a cada uno de sus almacenes mediante trenes de carga u otros medios, de forma que podemos escoger en qué almacén está cada paquete al inicio de la jornada. A su vez asumimos que, al inicio de la jornada, podemos tener cada uno de los vehículos en uno de nuestros almacenes a nuestra elección.

Una vez los paquetes se encuentran en un almacén en una ciudad, un vehículo puede pasar a recogerlos si tiene espacio. Se asume que todos los paquetes ocupan lo mismo, y que nuestros vehículos tienen capacidad para 2 (furgoneta) o 4 (camión) paquetes. De la misma forma, un vehículo puede entregar un paquete si se encuentra en la misma localidad donde debe entregarse. Tanto la acción de recoger como la acción de entregar toman el mismo tiempo. Asumimos que no tendremos que entregar más de un paquete a una misma dirección. También asumimos que el tiempo gastado en cargar un paquete es el mismo estemos ya en el almacén o teniendo que desplazarnos a él desde un sitio en la misma localidad, ya que con lo voluminosos que son los paquetes, la carga del paquete cuesta mucho más que un desplazamiento intra-localidad.

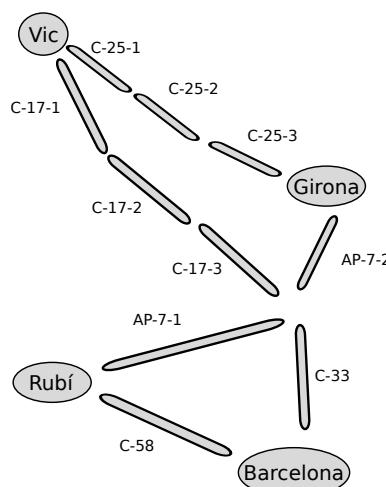
Los vehículos pueden moverse libremente entre los domicilios de una misma localidad. Sin embargo, para el desplazamiento entre dos localidades, deberá usar una serie de carreteras para desplazarse. Conocemos el mapa de carreteras de la región y la conectividad entre ciudades. Como dificultad añadida, las carreteras son de diferente longitud, y no es lo mismo recorrer una carretera larga que una corta. Para ello, un conjunto de expertos evalúa la longitud de cada carretera, y las ha dividido en 'tramos', de forma que el tiempo que se tarda en entregar un paquete en un domicilio, recogerlo de un almacén, y entre tramos (o entre un tramo y una localidad) es el mismo.

- Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Ten en cuenta que el modelo del dominio ha de poderse extender a un número arbitrario de paquetes, vehículos, localidades y carreteras.
- La empresa nos proporciona un ejemplo de prueba para saber como es la entrada del sistema. Describe el problema usando PDDL. Da una breve explicación de cómo modelas el problema. No es necesario escribir el init completo siempre que sea obvio qué falta, dando al menos un ejemplo de uso de cada predicado.

Paquete	Domicilio, localidad
P1	Av. Diagonal 3, Barcelona
P2	Av. Paral · lel 26, Barcelona
P3	C/ Gavà 8, Barcelona
P4	P. Olot 4, Girona
P5	C/ Joan Alsina 7, Girona
P6	C/ Torelló 2, Vic
P7	Av. Can Sucarrats, Rubí

Vehículo	Tipo
C1	Camión
F1	Furgoneta

Almacén	Localidad
A1	Barcelona
A2	Girona



Mapa de carreteras de la región, dividido en tramos

6. Hace años la burbuja inmobiliaria y de infraestructuras junto a los delirios de grandeza de algunos políticos provocó la creación de múltiples aeropuertos distribuidos por la geografía española. Sin embargo la crisis económica ha obligado ahora al Gobierno a tomar medidas y exigir a estos aeropuertos un plan de viabilidad para evitar su cierre. Por eso los gestores de un pequeño aeropuerto de la costa Mediterránea nos piden participar en un proyecto piloto de automatización de la torre de control del aeropuerto usando técnicas de inteligencia artificial. En concreto nos piden un sistema automático que, cada media hora, recibe la lista de vuelos que están pendientes de aterrizar o despegar del aeropuerto (y en el caso de los pendientes de despegue, la puerta de embarque que ocupan) y ha de generar un plan que diga en qué orden han de despegar o aterrizar los aviones (y en el caso de los que aterrizan, en qué puerta de embarque han de desembarcar).

El aeropuerto dispone de dos pequeñas terminales (*A* y *B*) con cinco puertas de embarque (*A1*, *A2*, *B1*, *B2* y *B3*). Dispone también de dos pistas para el despegue y aterrizaje de aviones (*ps1* y *ps2*), pero los fuertes vientos de la zona hacen que a veces no se pueda usar una de ellas. Nunca puede haber dos aviones a la vez en la misma pista (por ejemplo un avión no puede aterrizar si hay otro avión que acaba de aterrizar o un avión a punto de despegar; un avión no puede entrar en la pista para despegar si hay otro a punto de despegar o un avión aterrizando que no ha salido de la pista). Como las pistas están muy cerca de las puertas de embarque y hay poco espacio para maniobrar, por motivos de seguridad no se permite a un avión en tierra desconectarse de una puerta de embarque e ir a una pista si la pista no está libre de aviones, ni a un avión que acaba de aterrizar dirigirse a una puerta de embarque si esta tiene un avión).

Para simplificar el problema y no tener que modelar tiempos de despegue, aterrizaje o desplazamiento entre pistas y puertas de embarque, aprovechando que el aeropuerto es tan pequeño (y que se han de observar unas distancias de seguridad entre aviones) consideraremos que un avión que ha de despegar solo puede estar o bien en la puerta de embarque, o bien en la pista a punto de despegar o bien ya ha despegado. De igual forma un avión que ha de aterrizar en el aeropuerto sólo puede estar o bien pendiente de aterrizar, aterrizado ya en la pista o en la puerta de embarque para desembarcar. No hace falta modelar los estados intermedios y solo se puede pasar de un estado al siguiente si la pista o la puerta de embarque a la que se ha de mover el avión no tiene otro avión ocupándola.

- (a) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Ten en cuenta que el modelo del dominio ha de poderse extender no sólo a más o menos aviones sinó también a más o menos pistas y a más o menos puertas de embarque.
- (b) El aeropuerto nos ha proporcionado el listado de salidas y llegadas programadas para la próxima media hora. También nos avisa que sólo tenemos en estos momentos la pista *ps2* operativa (un fuerte viento lateral hace impracticable usar la otra pista en los próximos minutos). Nos pide que el planificador genere un plan de salidas y llegadas para la próxima media hora en la que se asigne un orden entre los despegues y aterrizajes (no ha de seguir el orden en el que aparecen en la tabla) y se asigne una puerta de embarque para los vuelos que aterrizan.

SALIDAS		
<i>vuelo</i>	<i>destino</i>	<i>puerta</i>
IB0051	Madrid	A1
BA6136	Londres	B1
AF0700	Lyon	B2
AL8860	Milán	B3

LLEGADAS		
<i>vuelo</i>	<i>procedente de</i>	<i>puerta</i>
IB0121	Málaga	—
VY0256	Barcelona	—
KL0333	Amsterdam	—
LH4044	Berlín	—

Describe este problema usando PDDL. Da una breve explicación de cómo modelas el problema.

7. La empresa *TraeMeLo* quiere convertirse en la líder dentro del sector emergente de empresas que reciben pedidos de sus clientes para ir a recoger algún producto/compra en uno de los comercios registrados en el sistema (normalmente una tienda, una farmacia, un restaurante) y traérselo a sus casas, siempre que sea dentro de la misma ciudad, no exceda los 10kg de peso y con un volumen de

máximo 40x40x30cm. Los pedidos los llevan unos mensajeros, que pueden ir en bici (con un cajón trasero en el que pueden llevar un máximo de 2 pedidos a la vez) o en scooter (con un cajón trasero en el que pueden llevar un máximo de 4 pedidos a la vez).

Los clientes pueden solicitar a través de una app la recogida de un pedido en un comercio registrado (la app envia un identificador de comercio, y el sistema tiene un método auxiliar para convertir ese string en coordenadas de GPS) y la entrega en la dirección registrada del cliente (hay que tener en cuenta que un cliente puede realizar más de un pedido, y en una misma dirección puede haber más de un cliente). El sistema entonces queda a la espera a que el comercio envie un aviso de que el pedido está listo para ser recogido, y en ese momento asigna alguno de los mensajeros que tenga espacio dentro del cajón trasero para recoger el pedido en el comercio y llevarlo a su destino, pudiendo parar por el camino en otros puntos para recoger o entregar pedidos. La empresa nos pide que desarrollemos un planificador simple que genere un plan para recoger y entregar los pedidos recibidos en un periodo de tiempo con los mensajeros disponibles en cada momento (a veces una avería o una baja por enfermedad hace que ese número varie). En el modelo deben quedar explícitos los pedidos a recoger, el aviso del comercio de que el pedido está listo, la recogida del pedido por parte del mensajero y su entrega al cliente.

- (3.5 puntos) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Ten en cuenta que el modelo del dominio ha de poderse extender a más o menos mensajeros, más o menos comercios, más o menos clientes y a futuros vehículos capaces de llevar más pedidos a la vez.
- (1.5 puntos) El jefe del proyecto nos ha proporcionado una tabla como ejemplo de las peticiones que el sistema recibirá en un periodo de 20 minutos, con una serie de pedidos que se han de transportar desde un comercio registrado hasta la dirección del cliente. También nos dice que al inicio de ese intervalo de 20 minutos tenemos disponibles dos motos (capacidad máxima de 4 pedidos) y una bici (capacidad máxima de 2 pedidos) estacionadas en el parking central de *TraeMeLo*, y que tras entregar todos los pedidos del periodo las motos y la bici han de volver a ese parking.

pedido	dirección recogida	dirección entrega	cliente
p1	PizzaHotBalmes88	PauClaris55	Mario
p2	TodoPastaAragon167	Meridiana131	Aitana
p3	SupermercadoBruc31	Villaroel21	Pau
p4	frankfurtCasp13	ManuelGirona86	Amalia
p5	MercahoyValencia216	RamblaPrim12	Jaume
p6	Farmacia24hDiagonal340	Meridiana131	Aitana
p7	TodoPastaAragon167	Lepant95	Nuria
p8	FarmaciaGarciaSardenya50	Lepant95	David
p9	PizzaHotBalmes88	RocBoronat01	Esther
p10	BurgerFastCiencies62	AvCarrilet73	Miquel
p11	Farmacia24hDiagonal340	AvCarrilet73	Miquel
p12	MercahoyValencia216	RamblaPrim12	Laura

Describe este problema usando PDDL. Da una breve explicación de cómo modelas el problema.

- Las redes de distribución de contenidos (CDNs en inglés) son redes de ordenadores que se colocan entre los usuarios y los proveedores de contenido (por ejemplo páginas web) para hacer caching de los contenidos. Con esto se consigue aligerar la carga de los proveedores y reducir la latencia de las peticiones de los usuarios.

Queremos utilizar planificación automática para gestionar el funcionamiento de una red CDN, y usaremos un modelo muy simplificado del funcionamiento real de este tipo de sistemas. Como escenario, consideraremos que tenemos un conjunto de usuarios haciendo peticiones a un conjunto de proveedores de contenido y que tenemos en medio una red de distribución formada por un conjunto de ordenadores. Cada nodo (ordenador) de la red de distribución puede almacenar un número determinado (que puede variar entre nodos) de contenidos en su caché.

Cada proveedor de contenido almacena o produce un subconjunto de contenidos que puede ofrecer, y consideraremos que al inicio de la ejecución del proceso de planificación tenemos como entrada un conjunto de peticiones realizadas por los usuarios. Cada petición especifica el contenido que debe ser entregado al usuario y puede ser resuelta únicamente por un nodo de la red de distribución. El mismo contenido puede ser solicitado por más de un usuario.

En este contexto simplificado, la comunicación entre los diferentes componentes del sistema se regiría por el siguiente protocolo:

1. El cliente realiza una petición por un contenido.
2. Un nodo de la red CDN procesa esta petición y entrega el contenido al cliente.

Es decir, que no hay comunicación directa entre el cliente y los proveedores. Sin embargo, otro aspecto importante es decidir cuál de los nodos será el que intercepte la petición y se encargue de entregar el contenido al cliente. Si un nodo de la red de distribución tiene en su caché el contenido especificado por una petición no resuelta, este mismo nodo puede entregar los contenidos directamente al usuario. En caso de que no haya ningún nodo con ese contenido, cualquier nodo con alguna posición libre en su caché realizará una petición al proveedor de ese contenido. El proveedor debe responder a esta petición, tras lo cual inmediatamente el nodo entrega el contenido al cliente y lo guarda en su caché al mismo tiempo. Si no hay ningún nodo que pueda satisfacer la petición de un usuario y las cachés de todos los nodos están llenas, cualquier nodo puede liberar una posición cualquiera de su caché. En resumen:

1. En primer lugar, se escoge un nodo cualquiera (y sólo uno) que tenga el contenido en cuestión en su caché, y este nodo entrega el contenido al usuario.
2. En caso de que 1. no se pueda cumplir, se escoge un nodo cualquiera (y sólo uno) que tenga alguna posición libre en su caché.
3. En caso de que ni 1. ni 2. se cumplan, se escoge un nodo cualquiera (y sólo uno) y éste liberará una posición cualquiera de su caché para atender a la petición.

Tanto en 2. como en 3. el nodo de la red CDN debe obtener el contenido. Para ello deberá lanzar una petición a algún proveedor que ofrezca ese contenido, recibir el contenido del proveedor, guardarlo en una posición de su caché y entregar el contenido al usuario.

Queremos implementar un sistema que permita ejecutar este proceso, tomando como punto de partida las peticiones creadas, los nodos y proveedores existentes, qué contenidos tienen los proveedores y cuántas posiciones de caché tiene cada nodo. El resultado es un plan que lista el orden en el que se realizan las acciones necesarias para atender a todas las peticiones cumpliendo con los protocolos especificados en los párrafos anteriores.

Nos interesa que el sistema registre, de manera explícita, los hits de caché, las peticiones de contenido de los nodos a los proveedores, las respuestas de los proveedores a los nodos, y las entregas de contenido a los usuarios.

- (a) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Ten en cuenta que el modelo del dominio ha de poderse extender a cantidades arbitrarias de clientes, nodos, proveedores, contenidos y peticiones de contenido.
- (b) Para probar el sistema nos dan el siguiente ejemplo de configuración:
  - 3 clientes: C1, C2, C3.
  - 2 nodos CDN con 3 posiciones caché disponibles y vacías: N1, N2.
  - 8 contenidos: O1, O2, O3, O4, O5, O6, O7 y O8.
  - 4 proveedores: V1 (que provee O1 y O2), V2 (O3, O4, O5), V3 (O6, O7) y V4 (O8).
  - Las siguientes peticiones iniciales:

<i>petición</i>	<i>cliente</i>	<i>contenido</i>
P1	C1	O1
P2	C2	O1
P3	C1	O2
P4	C3	O3
P5	C2	O4
P6	C2	O5
P7	C2	O6
P8	C3	O7
P9	C1	O8
P10	C1	O1

Describe este problema usando PDDL. Da una breve explicación de cómo modelas el problema.

9. El mago Rincewind debe componer un hechizo a partir de conocimiento arcano que se encuentra en diversas torres esparcidas por el mundo. El hechizo está compuesto por un conjunto de thaums, cada uno de tipo distinto. En cada torre hay exactamente un único tipo. Para completar un hechizo se requiere de tener un thaum de cada uno de los tipos que requiere el hechizo.

Sin embargo, las torres pueden estar protegidas por encantamientos potentes (máximo de uno por torre), y sólo aquellos que tienen la contramedida a ese encantamiento pueden entrar. Por ejemplo, un encantamiento lanza-flechas o estruja-magos requiere de tener un hechizo de escudo físico, o un encantamiento de luz solar requiere de un encantamiento de crema de protección solar. Para cada torre, sabemos qué contramedida (única) se necesita para deshabilitar su encantamiento, si es que se necesita una contramedida. Sin la contramedida apropiada, Rincewind no se atreve a intentar entrar en la torre. Las contramedidas se pueden reutilizar tantas veces como se necesite, no se gastan.

Rincewind no es el más apto de los magos, y no empieza sabiendo todas las contramedidas necesarias. Sin embargo, dentro de las torres puede haber otras contramedidas que Rincewind puede aprender para proseguir su camino.

- (a) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Se valorará positivamente que el modelo del dominio pueda extenderse a cantidades arbitrarias de tipos de thaums, contramedidas y torres. Para obtener puntuación completa, se debe modelar de forma que pueda minimizar la distancia recorrida en el camino entre torres.
- (b) Para probar el sistema, intentamos hacer un hechizo sencillo con 5 thaums: arriba, abajo, lado, seductor y hierbabuena. Hay 15 torres en el mapa. Las torres 1 a 5 están desprotegidas. Las otras 10 tienen protección mágica variada. Las torres contienen los 5 thaums objetivo y otros 3 (arcana, agave y bourbon) que no nos interesan.

Torre	Thaum	Contramedidas (en torre)	Trampa	Contramedida (para trampa)
t1	arriba	paraguas, sombrero puntiagudo	-	-
t2	hierbabuena	sombrero puntiagudo	-	-
...				
t5	arcana	-	-	-
t6	agave	crema solar	Estruja-magos	Escudo físico
t7	bourbon	Matasuegras mágico	Lluvia torrencial	Paraguas
...				
t15	seductor	paraguas	Luz solar	crema solar

Rincewind empieza con contramedidas de escudo físico y caramelos de menta, en una torre 't0' sin nada (no hay thaums ni contramedidas a aprender).

Describe el problema usando PDDL. No es necesario escribir el init completo siempre que sea obvio qué falta, dando al menos un ejemplo de uso de cada predicado. Podéis inventar cualquier información que no esté detallada en la tabla (como distancias o torres que no están explícitas).

10. La FIB nos ha pedido una herramienta sencilla en que los alumnos puedan decir que asignaturas quieren cursar en los últimos semestres de la carrera y les haga un plan de matriculación que, basándose en las asignaturas que ya tienen cursadas, les diga de que asignaturas se han de matricular como mínimo los próximos semestres para poder llegar a matricularse de las asignaturas que no se desean perder.

El sistema tendrá conocimiento sobre las asignaturas del plan de estudios, sus pre-requisitos, sus pre-corequisitos (si los hay), las asignaturas que el alumno ya ha cursado y las asignaturas que el alumno quiere cursar. El resultado es un plan de matriculación que 1) refleja las asignaturas mínimas que el alumno ha de matricular, 2) para cada asignatura, indica en qué semestre ha de hacerla, 3) para todas las asignaturas del plan se cumplen en todo momento los pre-requisitos y/o pre-corequisitos necesarios para cursarlas

Para simplificar el problema, 1) no pondremos un límite de créditos matriculables por semestre, 2) del sistema de pre/co/pre-corequisitos solo modelaremos dos tipos de requisitos entre asignaturas

- pre-requisito: Si una asignatura A es **prerequisito** de una asignatura B, para poder matricularse de B se ha de haber cursado la asignatura A en un semestre anterior. Ejemplos de prerequisitos: PRO1⇒BD, PRO1⇒EDA, PRO2⇒BD, PRO2⇒EDA, BD⇒IES, IES⇒AS, AS⇒ECSDI, M1⇒PE, M2⇒PE, EC⇒SO, IC⇒SO, A⇒AA, EDA⇒IDI, EDA⇒IA, EDA⇒PROP, EDA⇒LI, EDA⇒TC, EDA⇒PAR, SO⇒PAR
  - pre-corequisito: Si una asignatura A es **pre-corequisito** de una asignatura B, para poder matricularse de B se ha de haber cursado la asignatura A en un semestre anterior o matricularla en el mismo semestre que A (es decir, matricular A y B a la vez). Ejemplos de pre-corequisitos: BD⇒PROP, PE⇒A, PROP⇒IA, PROP⇒LI, PROP⇒TC, PROP⇒A, PE⇒A
- (a) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido.
- (b) Como primera prueba del sistema nos dan un alumno de la FIB que solo ha aprobado PRO1, PRO2, EC, IES y EDA. Nos pide que el planificador genere un plan de matriculación en el que pueda llegar a cursar las asignaturas ECSDI, AA y PAR. Describe este problema usando PDDL. Da una breve explicación de cómo modelas el problema. No es necesario escribir el init completo siempre que sea obvio qué falta, dando al menos un ejemplo de uso de cada predicado.
11. El mago Rincewind, tras retirarse de sus aventuras, ha creado un gremio inteligente de fabricación de regentes y artilugios mágicos. Este gremio es un centro generalista, encargado de la generación de prototipos a partir de pasos, organizando su producción entre diversas máquinas y montando planes de producción. Sin embargo, el encontrar estos planes a mano es bastante tedioso, así que ha pedido a sus becarios un sistema que se encargue de planificar la producción.

Cuando un problema llega al gremio, el objetivo es la secuencia de pasos y máquinas que se deben usar para fabricarlo. Para cada componente o artilugio sabemos qué máquina puede usarse para fabricarlo y qué productos necesita, que dependen de cada problema específico. Un componente o artilugio puede ser fabricado de muchas formas diferentes (ej. se puede conseguir sustrato con centrifugadora:suero+bilis de basilisco, o con materializador:arcana o tienda:-), pero podéis asumir que no hay dos formas diferentes de hacerlo con la misma máquina. Cada componente requiere siempre de una máquina, pero puede requerir de un número de ingredientes arbitrario. Por falta de almacenamiento, podéis asumir que nunca habrá más de un ingrediente del mismo tipo (ej. no puede haber dos sueros a la vez).

El problema tiene una complejidad adicional: la fabricación tiene una duración. Esto quiere decir que meter los productos en la máquina no implica tener el resultado inmediatamente. Asumimos que todas las máquinas tardan exactamente el mismo tiempo en generar cualquier componente, y que el tiempo de cargar las máquinas de ingredientes es despreciable. Sin embargo, una máquina no puede hacer dos componentes a la vez. Queremos minimizar el tiempo que tardamos en tener todos los productos pedidos en el problema al gremio.

- (a) Describe el dominio (incluyendo predicados, acciones, etc.) usando PDDL. La puntuación será proporcional a cuánto del problema se ha podido modelar (eg. poder generar productos < generar

productos con un número de ingredientes variable < que las máquinas tarden en producir < minimizar tiempo de producción).

- (b) El gremio ha recibido una petición de hacer un talismán: ‘patito de goma aprueba-exámenes’, seguramente de uno de los becarios. El gremio dispone ahora mismo de madera y plástico. Debajo se listan los procesos de fabricación posibles. Representa el problema en PDDL acorde con vuestra propuesta de dominio del apartado anterior.

Fabricado	Máquina	Ingredientes
Patito	M.Ensamblaje	Pico, cuerpo, ojos
Cuerpo	Soplador	Plástico, molde
Plástico	Tienda	-
Plástico	Materializador	-
Molde	Fundición	Plástico
Molde	CNC	Madera
Pico	Simbolificador	Pico real
Pico real	Descuartizapatos2000	Pato
Ojos	Simbolificador	Espíritu, plástico
Espíritu	Materializador	-
Pato	Materializador	-
Pato	Tienda	-

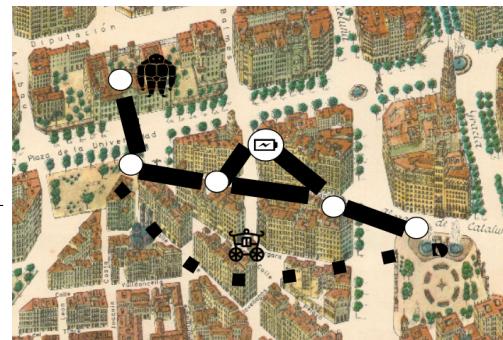
12. El mago Rincewind debe ayudar con el diseño cognitivo de un Golem mensajero. Un Golem es un artilugio que se alimenta de una batería arcana (que tiene una carga) para desplazarse y hacer diversas tareas. Esta se consume a un ritmo dependiente de las acciones que el Golem toma: por ejemplo, desplazarse 100 metros (tamaño exacto de un desplazamiento estándar) son 20 unidades de carga, entregar un mensaje depende de la longitud de este, etc. El apartado b) determina los costes para el modelo actual diseñado por el departamento arcanomecánico. Además, el Golem dispone de una cantidad de oro que puede usar para recargar su batería (1 oro recarga la batería a su máximo independientemente de la carga actual) en localizaciones específicas.

Además, un Golem es especialmente lento en desplazamiento (por lo que Rincewind considera este proyecto un gasto terrible de fondos públicos). Por ello, Rincewind ha decidido también otorgar al golem la capacidad de pagar por el uso de carruajes para el desplazamiento. Los carruajes tienen un coste (en oro) asociado a sus posibles desplazamientos. Los carruajes solo pueden desplazarse por carreteras, así que no se pueden utilizar, por ejemplo, dentro de un edificio. Su accesibilidad viene determinada como parte del problema especificado. Para usar un carruaje, se debe llamar y subir a uno, desplazarse y pagar y finalmente bajar en tu destino.

El objetivo del diseño cognitivo del que Rincewind está encargado es determinar la secuencia de acciones a realizar para que el Golem entregue su mensaje al destinatario. El problema viene descrito como un mensaje (con un coste en batería por entregar), una dirección, batería (con una carga máxima determinada) totalmente cargada, oro inicial y un mapa con información sobre rutas de carruajes y accesibilidad entre calles. En el plan del Golem deberá respetar las restricciones sobre oro y batería (como no pagar más oro del que tiene, o no hacer acciones cuyo coste en batería excede su batería actual). El plan del Golem debería detallar los desplazamientos que hace a pie (de localización en localización), subir a un carruaje, desplazarse en carruaje, bajar de un carruaje, cargar su batería en un punto de carga, y la entrega del mensaje final.

- (a) Describe el dominio (incluyendo predicados, acciones, etc.) usando PDDL. Tened en cuenta que los detalles del apartado b) dependen de la implementación del Golem y el dominio debe poder adaptarse a cambios sobre esta.
- (b) Representa el fichero de problema para la siguiente especificación de Golem (Versión A), teniendo que entregar el mensaje ‘Ook. Ook.’ desde la Biblioteca de la UI, a Plaza Tarraconensis. Cada desplazamiento descrito en la 2<sup>a</sup> tabla es de exactamente 100 metros, las entradas en cursiva indican un lugar de recarga. La imagen es una representación visual del mapa:

VA-Golem	Unidades
Carga Máxima	100 E
Oro inicial	20 o
Coste mensaje	10 E
Coste Subir Carruaje	3 E
Coste Bajar Carruaje	2 E
Carruaje Origen	Destino (coste)
Universidad Invisible	Plaza Tarraconensis (10 o)
Plaza Tarraconensis	Universidad Invisible (10 o)



Lugar Origen	Accesible Andando
Biblioteca UI	Universidad Invisible
Universidad Invisible	Biblioteca UI, Ronda Universidad 1
Ronda Universidad 1	Universidad Invisible, Ronda Universidad 2, Puesto de carga
<i>Puesto de carga</i>	Ronda Universidad 1, Ronda Universidad 2
Ronda Universidad 2	Ronda Universidad 1, Puesto de carga, Plaza Tarraconensis
Plaza Tarraconensis	Ronda Universidad 2

13. Una empresa española ha decidido entrar en el sector de los coches autónomos poco a poco. Bajo el nombre de *GoAuto* ha desarrollado tres tipos de vehículo eléctrico totalmente autónomos que no tienen volante ni otro tipo de mecanismos de manejo manual: el *GoAuto2*, vehículo de 2 plazas similar al de Google, el *GoAuto4* con similares características y 4 plazas, y el *GoAuto7* con 7 plazas. Para ayudar al desarrollo de la tecnología la empresa ha creado un proyecto piloto de movilidad urbana autónoma personalizada en las calles de una pequeña ciudad para substituir a los taxis y a los micro-buses. Se ha creado una red de 15 *GoPuntos*, puntos de recogida que substituyen a las antiguas paradas de bus y taxi de la población, con la idea de crear más puntos en el futuro. Los ciudadanos pueden pedir a través de una app que uno de estos vehículos los lleve desde un punto de recogida a otro punto en la ciudad. Alguno de los vehículos autónomos que tenga plazas libres recogerá al ciudadano en el punto de recogida y lo llevará al punto de destino, pudiendo parar por el camino en otros puntos para recoger o descargar pasajeros. La empresa nos pide que desarrollemos un planificador simple que genere un plan para servir las peticiones de movilidad de los ciudadanos con los vehículos autónomos disponibles en cada momento (al ser prototipos a veces falla alguno y se han de servir las peticiones con el resto).

- (a) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Ten en cuenta que el modelo del dominio ha de poderse extender a más o menos *GoPuntos* y más o menos *GoAutos*.
- (b) El jefe del proyecto nos ha proporcionado una tabla como ejemplo de las peticiones de movilidad que el sistema recibe cada 15 minutos, con una serie de ciudadanos que se han de transportar entre los 15 puntos de la red (*GoPunto1*, *GoPunto2* ... *GoPunto15*). También nos dice que al inicio de ese intervalo de 15 minutos tenemos un vehículo *GoAuto2* de 2 pasajeros vacío estacionado en *GoPunto3* y un vehículo *GoAuto4* de 4 pasajeros vacío estacionado en *GoPunto7*, y que el vehículo *GoAuto7* no está disponible por avería. Nos pide que el planificador genere un plan en el que los vehículos disponibles acaben llevando todos los ciudadanos a sus puntos de destino.

ciudadano	GoPunto origen	GoPunto destino
Julian	GoPunto1	GoPunto3
Paula	GoPunto7	GoPunto12
David	GoPunto14	GoPunto10
Eva	GoPunto3	GoPunto5
Laura	GoPunto3	GoPunto7
Felipe	GoPunto3	GoPunto8
Amelia	GoPunto1	GoPunto5
Ivan	GoPunto13	GoPunto8
Daniela	GoPunto6	GoPunto11
Hugo	GoPunto12	GoPunto10

Describe este problema usando PDDL. Da una breve explicación de cómo modelas el problema.

14. La cirugía ha avanzado tanto en los últimos años que, para las intervenciones menores, los quirófanos han adaptado su funcionamiento al sistema de producción en cadena, pudiendo tratar muchos pacientes cada hora. Un ejemplo de ello son las operaciones de cataratas. El paciente llega al hospital, se le hace pasar a un vestidor donde se quita la ropa de calle y se pone una bata, luego de allí pasa a una camilla donde se le aplica anestesia local en la superficie del ojo, se opera en tan solo unos minutos, se deja al paciente descansar unos minutos para que se recupere de la anestesia, y después el paciente vuelve al vestidor a vestirse y sale por su propio pie del hospital, con un vendaje protegiendo el ojo.

Un centro de referencia en cirugía ocular nos ha pedido un sistema planificador que establezca un orden correcto para los diferentes pasos del proceso durante una hora, evitando bloqueos. El sistema recibe la lista de pacientes a tratar e irá simulando como van pasando de un paso a otro. Para simplificar el modelo, todos los pasos tienen la misma duración (entrada y registro, desvestirse, anestesia y operación, recuperación, vestirse , salir), y solo se han de modelar cuatro espacios: la sala de entrada, los vestidores, el quirófano con sus camillas y la sala de salida. Tanto la sala de entrada como la de salida tienen capacidad de sobra para todos los pacientes de una hora. En el caso de los vestidores son pequeñas habitaciones donde el paciente se quita la ropa y se pone la bata de quirófano, antes de la operacion, y tras ella se quita la bata y vuelve a vestirse con la ropa de calle. Cada vestidor tiene capacidad para una sola persona a la vez y normalmente hay menos vestidores que camillas, por lo que se ha de controlar que en cada momento haya como máximo una sola persona por vestidor. Tras desvestirse el paciente solo podrá entrar a quirófano cuando haya una camilla libre. En esa camilla se mantendrá durante la anestesia, operación y recuperación (son los médicos los que se van moviendo de una camilla a otra para operar cuando el paciente está listo). Cuando el paciente se recupera de la anestesia, ha de esperar a que haya un vestidor libre para volver a vestirse y marchar. Hay que tener cuidado en este punto, ya que se podrían crear bloqueos si tenemos todos los vestidores ocupados con pacientes esperando operación y todas las camillas del quirófano ocupadas con pacientes que querrán volver a vestirse.

- (a) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Ten en cuenta que el modelo del dominio ha de poderse extender a más o menos vestidores y a más o menos camillas dentro del quirófano. Supondremos que siempre hay un solo quirófano.
- (b) Para probar el sistema nos piden que modelemos un centro con una sala de entrada, tres vestidores, un quirófano con cuatro camillas y una sala de salida. Tenemos 7 pacientes por operar: Joan, Luis, Mireia, Alba, Ester, Pau i Quim. No hay orden de entrada prefijado entre estos pacientes, el planificador nos dirá quienes entran primero en los vestidores para desvestirse y los pasos posteriores hasta que todos los pacientes hayan sido operados y hayan salido del centro.

Describe este problema en PDDL acorde con tu propuesta de dominio del apartado anterior. Por favor incluye una corta explicación sobre como modelas el problema. No es necesario escribir el estado inicial completo siempre que sea obvio qué falta, dando al menos un ejemplo de instancia de cada predicado para que se vea como se usa. Si es necesario escribir el objetivo por completo.

15. Queremos desarrollar una app capaz de recomendar programas de entrenamiento físico al usuario del dispositivo. La app posee un conjunto de ejercicios que se pueden realizar dentro o fuera de un gimnasio, como ejercicios de suelo, ejercicios de estiramientos/flexibilidad, ejercicios con aparatos (bicicleta estática, cinta de andar, remo, stepper, pesos...), ejercicios con o sin pesas para los diferentes grupos musculares, correr, patinar, nadar, etc. Cada ejercicio tiene diez niveles de dificultad (del 1 al 10), donde el aumento de dificultad en un mismo ejercicio se consigue alargando su duración, incrementando su intensidad, el peso a mover, el número de series o reduciendo el tiempo de descanso entre series.

La interfaz de la app permite al usuario indicar los ejercicios que el usuario está realizando en este momento y su nivel de dificultad inicial (del 1 al 10), y el nivel de dificultad que quiere llegar a alcanzar para algunos de ellos, y la app le generará un plan de entrenamiento físico semanal especificando los ejercicios a realizar cada día y su orden, con el objetivo de conseguir subir de nivel en el conjunto de ejercicios escogidos. El plan de entrenamiento ha de tener en cuenta las siguientes dependencias y restricciones:

- ejercicios preparadores: son ejercicios que se han de hacer antes de un cierto ejercicio en el mismo día de entrenamiento pero no tienen porque ir inmediatamente antes (un ejemplo son los estiramientos, que preparan para varios ejercicios de la sesión de entrenamiento). Es una restricción estricta que no se puede romper.
- ejercicios nivel +1: sirve para modelar la evolución en el nivel de dificultad de un ejercicio según pasan los días. Para cada ejercicio podremos subir un nivel  $N + 1$  solo si ese ejercicio se ha realizado en algún día anterior en el nivel  $N$ .
- no repetir ejercicio el mismo día: en un día de entrenamiento no se debe repetir el mismo ejercicio
- límite de ejercicios por día: no se pueden colocar más de 6 ejercicios en un día de entrenamiento.

La app ya tiene cargada la lista completa de ejercicios, sus niveles y los ejercicios preparadores de un ejercicio. Del usuario recibe la lista de los ejercicios que ya está haciendo (y en qué nivel los hace) y los niveles incrementados de algunos de los ejercicios a los que quiere llegar al final de la semana. El resultado es un plan de entrenamiento que refleja, para cada día, la lista ordenada de los ejercicios que el usuario ha de hacer para llegar a los niveles objetivo cumpliendo las dependencias y restricciones, o un mensaje del planificador diciendo que no se puede llegar a esos niveles en solo 7 días. El planificador no tiene que llenar el plan con ejercicios extra, solo los necesarios para llegar al día 7 a los niveles deseados. Si sobran huecos se entiende que se le da al usuario libertad para hacer otros ejercicios suplementarios al plan de entrenamiento. En caso de que un usuario no especifique nivel inicial para un ejercicio el sistema asume que ha de empezar ese ejercicio en el nivel 1.

- (a) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Ten en cuenta que el modelo del dominio ha de poderse extender a más o menos días y más o menos ejercicios por día.
- (b) El jefe del proyecto nos ha proporcionado un ejemplo de uso del sistema, con los ejercicios que está haciendo y sus niveles objetivo.

PUNTO DE PARTIDA ejercicio (nivel actual)	OBJETIVOS DEL ENTRENAMIENTO: ejercicio (nivel final)
estiramientos (nivel 1)	
correr (nivel 1)	correr (nivel 4)
nadar (nivel 1)	nadar (nivel 4)
cuádriceps (nivel 1)	cuádriceps (nivel 4)
isquios (nivel 2)	
gemelos (nivel 3)	
abductores (nivel 2)	abductores (nivel 6)
remo (nivel 2)	remo (nivel 3)
dominadas (nivel 3)	dominadas (nivel 6)
pectoral (nivel 3)	pectoral (nivel 6)
bíceps (nivel 3)	bíceps (nivel 7)
tríceps (nivel 3)	tríceps (nivel 6)
dorsal (nivel 2)	dorsal (nivel 4)

También nos dan la siguientes listas de ejercicios preparadores:

- los estiramientos son ejercicio preparador de todos los demás;
- las sentadillas son ejercicio preparador de cuádriceps y correr;
- las dominadas son ejercicio preparador de pectoral;
- cuádriceps ejercicio preparador del de remo.

Describe este problema usando PDDL. Da una breve explicación de cómo modelas el problema.

16. La empresa *ComeBien* quiere un sistema inteligente que ayude a programar los menus semanales en su cadena de restaurantes.

Un menú esta formado por un primero y un segundo para cada día laborable (de lunes a viernes). Asumiremos que el cocinero es capaz de preparar una serie de primeros y segundos que podremos combinar en los menús. Cada plato tendrá asociado un tipo (sopa, crema, ensalada, carne, pescado, ...).

Se ha de tener en cuenta las incompatibilidades entre los primeros y segundos de manera que no pueden aparecer el mismo día del menú (por ejemplo ensaladilla de primero y fabada asturiana de segundo). También se quiere que no se utilice un mismo plato más de una vez a la semana.

También podemos tener restricciones en el tipo de plato (sopa, crema, ensalada, carne, pescado, ...) entre días consecutivos de manera que este no pueda repetirse. La restricción la observaremos para primeros y segundos separadamente de manera que no podemos poner dos días seguidos dos primeros platos de pescado (salmón un día y lubina el siguiente), pero puede haber un primero de pescado un día (paella) y un segundo de pescado al día siguiente (salmón).

Nos piden que el planificador genere el menu para una sola semana (primero y segundo, de lunes a viernes) sin repetir platos, sin incompatibilidades en el mismo día y cumpliendo las restricciones de tipo de plato en dias consecutivos. Ha de quedar claro para cada plato en que día se sirve y si es primero o segundo.

- Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Ten en cuenta que el modelo del dominio ha de poderse extender a más o menos platos, y más o menos incompatibilidades.
- El director de la area TIC de *ComeBien* nos ha proporcionado una lista de platos como ejemplo de la entrada que recibiría el planificador, y un listado de las incompatibilidades entre ellos. Se nos pide que el planificador genere un plan usando esta información.

PRIMEROS		SEGUNDOS	
plato	tipo	plato	tipo
ensalada_tomate	ensalada	fabada	carne
crema_calabaza	crema	filete_ternera	carne
ensaladilla	ensalada	plato_salmon	pescado
paella	pescado	plato_dorada	pescado
escalivada	ensalada	plato_pollo	carne
sopa_pollo	sopa	butifarra	carne

incompatibilidades	
PRIMERO	SEGUNDO
sopa_pollo	butifarra
paella	plato_salmon
crema_calabaza	fabada
ensaladilla	fabada
crema_calabaza	filete_ternera
crema_calabaza	plato_pollo

Describe este problema usando PDDL. Da una breve explicación de cómo modelas el problema.

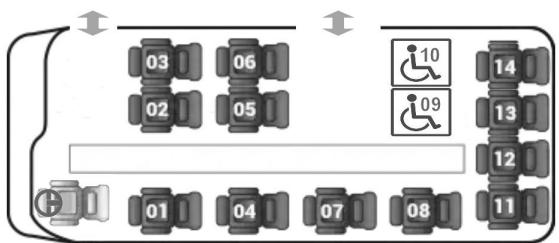
17. Una ciudad del este de Europa quiere posicionarse en el top 15 de las Smart Cities europeas incorporando la digitalización a la movilidad urbana. Después de reorganizar el transporte en el denso nucleo urbano, quieren solucionar ahora el problema de movilidad para todos los barrios periféricos (compuestos de pequeñas casas dispersas entre colinas, y donde no tiene sentido poner lineas de autobuses de recorrido fijo). Quieren crear el SmartBus, líneas de autobús de barrio donde el recorrido se adapta a los usuarios que haya en cada momento.

Cada barrio tendrá asignado uno (o más) autobuses de 14 plazas (12 asientos y 2 espacios reservados para personas de movilidad reducida). Por cada barrio se distribuyen tantas paradas del SmartBus como haga falta, y una o más paradas les permiten transbordar hacia otros medios de transporte del resto de la ciudad. Los vecinos del barrio pueden pedir, a través de una app, que un SmartBus los lleve desde una parada del barrio a otra. Uno de los autobuses que tenga plazas libres recogerá al viajero en la parada de origen y lo llevará a la parada de destino, pudiendo parar por el camino en otras paradas de SmartBus para recoger o descargar pasajeros.

La empresa nos pide que desarrollemos un planificador simple que genere un plan para servir las peticiones de movilidad de un barrio con los autobuses disponibles en cada momento (según la hora del día).

El resultado del plan ha de mostrar como va cada autobús de parada en parada, y para cada una de sus paradas 1) quien baja del autobús y 2) quien sube al autobús y que asiento tiene asignado.

- (a) (3,5 puntos) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Ten en cuenta que el modelo del dominio ha de poderse extender a más o menos paradas del SmartBus y más o menos autobuses.
- (b) (1,5 puntos) El jefe del proyecto nos ha proporcionado una tabla como ejemplo de las peticiones de movilidad que el sistema recibe cada 15 minutos, con una serie de ciudadanos que se han de transportar entre las 14 paradas del barrio (p2701, p2702 ... p2714) (los pasajeros con \* son de movilidad reducida y solo pueden usar las plazas 09 o 10). También nos dice que al inicio de ese intervalo de 15 minutos tenemos solo el autobús *SBus027*, vacío y estacionado en la parada p2701. Nos pide que el planificador genere un plan en el que el vehículo disponible acabe llevando todos los ciudadanos a sus paradas de destino.



pasajero	parada origen	parada destino	pasajero	parada origen	parada destino	pasajero	parada origen	parada destino
Esteban	p2701	p2703	Cristina	p2707	p2712	Alba*	p2703	p2707
Daniel	p2714	p2710	Luis	p2711	p2705	Beatriz	p2703	p2708
Lorena	p2703	p2707	Jesus	p2703	p2708	Jordi	p2706	p2705
Raul	p2701	p2705	Maria	p2713	p2708	Kilian*	p2713	p2708
Estela	p2706	p2711	Sebas*	p2712	p2710	Fabiola*	p2706	p2710
Martina	p2701	p2711	Noelia	p2707	p2705	Susana	p2714	p2710
Pau	p2704	p2710	Xavier*	p2703	p2705	Ruth	p2706	p2710

Describe este problema usando PDDL. Da una breve explicación de cómo modelas el problema. No hace falta incluir en el modelo toda la tabla de pasajeros con sus paradas de origen y destino, es suficiente modelar algunos. En vuestro modelo 1) debe haber al menos un ejemplo de uso de todos los predicados que deban ser inicializados en el fichero del problema, 2) la condición de fin si que ha de estar completamente modelada.

18. La empresa *LuxeCar* fabrica artesanalmente coches deportivos de alto nivel de calidad con un look clásico que está atrayendo a nuevos clientes. Para poder organizar las tareas dentro de su fábrica nos han pedido un sistema inteligente de control que les ayude a planificar como paralelizar los pasos que requieren los pedidos y gestionar correctamente los recursos compartidos (taller de pintura, taller de carrocería, taller de mecánica y pulidora de pintura).

Los pasos de producción para todos sus coches son, en este orden:

- crear carrocería - se crea una carrocería completa, sin pintar (requiere un taller de carrocería que esté libre)
- pintar carrocería - se pinta la carrocería (requiere de un taller de pintura libre)
- montar los elementos mecánicos - a la carrocería ya pintada se le añaden todos los elementos mecánicos (requiere un taller de mecánica que esté libre)
- montar los elementos del interior - se añaden todos los elementos del interior del coche y se conectan con la mecánica (requiere un taller de mecánica que esté libre)

Según el gusto del cliente hay tres pasos adicionales que pueden añadirse en algunos pedidos:

- montar kit sport - es un paso opcional que le añade un pack de elementos deportivos, tanto al exterior de la carrocería como en el interior del vehículo (este paso solo se puede hacer tras haber montado el interior del coche; requiere un taller de carrocería que esté libre)
- pulir pintura - es un paso opcional donde se pule la pintura en los casos en los que el cliente quiere un acabado extrabrilante (se puede insertar este paso en cualquier momento, siempre que el coche esté pintado y que, en caso de que el cliente haya pedido un kit sport, este ya esté montado; requiere de un taller de pintura libre y de una pulidora)
- crear certificado - solo si lo pide el cliente se crea un certificado de autenticidad para el vehículo de ese pedido (se puede insertar este paso en cualquier momento, no requiere recursos compartidos, se coge el certificado ya impreso en un papel especial con detalles en oro y se escriben a mano los datos del vehículo)

Es importante observar que tenemos pasos que son totalmente independientes del resto (crear carrocería y crear certificado) y pasos que dependen de que uno o más pasos en el pedido estén hechos ya para poder iniciarse. De cada recurso compartido puede haber uno o más de uno en la fábrica. Cada uno de los talleres tiene el personal calificado (que no se comparte con otros talleres o lugares de la fábrica) y dispone de todas las herramientas y todas las piezas necesarias para su actividad, pero hay algunas herramientas como la (o las) pulidora(s) que al usarse menos se han de compartir. Para los recursos compartidos (ya sean talleres o herramientas) nos piden que se modele explicitamente las acciones de asignar y liberar ese recurso, evitando que el sistema recomiende cosas incompatibles como dos pedidos

que usan la pulidora a la vez o un pedido montando una carrocería y otro pedido añadiendo un kit deportivo en el mismo taller de carrocería.

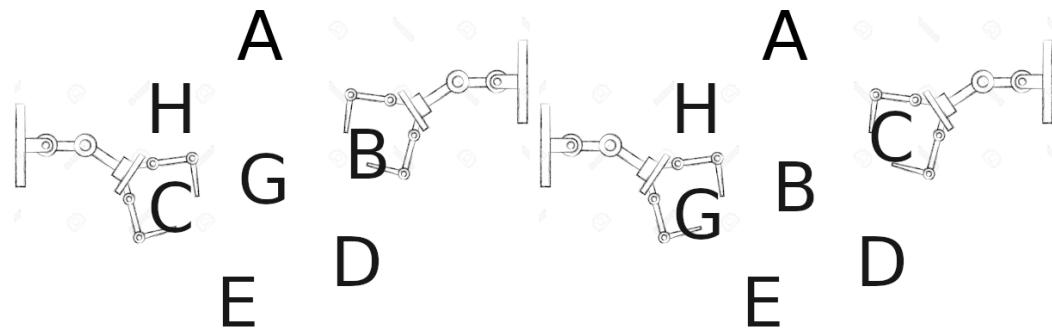
En el momento de instalar el sistema, la empresa ha de realizar un inventario de los recursos compartidos que hay. Y luego cada día ha de introducir los pedidos a realizar en ese día, diciendo para cada pedido qué pasos se han de realizar. El resultado es un plan que lista el orden en que han de realizarse los pasos para servir los pedidos del día, evitando conflictos. Solo se pide un orden de las tareas, no se tiene que modelar su duración o hacer una planificación minuto a minuto.

- (a) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Ten en cuenta que el modelo del dominio ha de poderse extender no sólo a más o menos pedidos por día sinó también a más o menos recursos compartidos (por ejemplo 4 talleres de pintura y dos pulidoras).
- (b) Para probar el sistema nos dan el caso particular de una de las posibles configuraciones que se están planteando para la fábrica, con un taller de carrocería, dos talleres de pintura, una máquina pulidora (a compartir por los dos talleres de pintura) y un taller de mecánica. Y nos dan como datos de prueba una posible lista de pedidos para un día dado con los pasos a hacer:

<i>pedido</i>	<i>crear carroceria</i>	<i>pintar carroceria</i>	<i>pulir pintura</i>	<i>montar mecánica</i>	<i>montar interior</i>	<i>montar kit sport</i>	<i>crear certificado</i>
P1	X	X		X	X		X
P2	X	X	X	X	X	X	X
P3	X	X	X	X	X		
P4	X	X		X	X		X
P5	X	X		X	X	X	
P6	X	X	X	X	X		
P7	X	X		X	X		X

Describe este problema usando PDDL. Da una breve explicación de cómo modelas el problema.

19. Un artista ha decidido incluir un componente de inteligencia artificial y tecnología en una exposición de la galería de arte moderno. Pretende incluir en su obra una disposición de piedras de colores rojo, verde y azul dispuestas en un círculo, con una piedra adicional en el centro. Puede haber más de una piedra de un mismo color. Además, dispone de dos brazos mecánicos que cuelgan del techo sobre las posiciones de las piedras del círculo. Ambos brazos no podrán estar jamás sobre la misma piedra al mismo tiempo. Cada brazo mecánico puede desplazarse de una posición del círculo a otra sin restricciones (aparte de la que acabamos de mencionar). El sistema de brazos mecánicos puede mover piedras haciendo rotaciones entre tres elementos las dos piedras de debajo de los brazos y la central. Por ejemplo, si las piedras de debajo de los brazos son C y B, y la central es G, denotamos el estado inicial como CGB, y podríamos rotarlo a GBC o BCG. De esta manera, en el caso de la rotación de CGB a GBC, la piedra central (G) acabará en una de las posiciones (C), la piedra que estaba en esa posición (C) irá a la posición donde estaba la otra piedra (B), y esta última (B) irá al centro. El objetivo final de mover estas piedras será cumplir una propiedad que tiene que ver con sus colores y su colocación en el círculo (exclusivamente qué puede haber inmediatamente delante o detrás), y que puede cambiar con cada problema (en la exposición cambiarán constantemente). Este es un ejemplo de rotación (entre C, G y B):



- (a) Describe el dominio (incluyendo predicados, acciones, etc...) usando PDDL. Da una explicación razonada de los elementos que has escogido. Ten en cuenta que el modelo del dominio ha de poderse extender a cantidades arbitrarias de posiciones en el círculo y colores.
- (b) Para probar el sistema nos dan el siguiente ejemplo de configuración En el círculo hay (en orden de las agujas del reloj) tres piedras rojas, tres azules y tres verdes, en el centro hay una piedra verde, los brazos están en la primera y última posición (encima de la primera roja y la última verde) y el objetivo final es que inmediatamente antes (en el orden de las agujas del reloj) de una piedra roja siempre haya una verde, y que no puede haber dos piedras azules seguidas.
20. El mago Rincewind debe crear un hechizo colocando  $N$  nexos arcanos en una cuadrícula de  $N \times N$  posiciones. Para que el hechizo funcione, es necesario que los  $N$  nexos estén en armonía, lo cual requiere que no comparten fila, columna ni diagonales.
- (a) Describe el dominio (incluyendo predicados, acciones, etc.) usando PDDL. Se valorará especialmente que la descripción sea eficiente a nivel temporal.
- (b) Representa el fichero de problema para la colocación de 5 nexos en PDDL acorde con vuestra propuesta de dominio del apartado anterior.