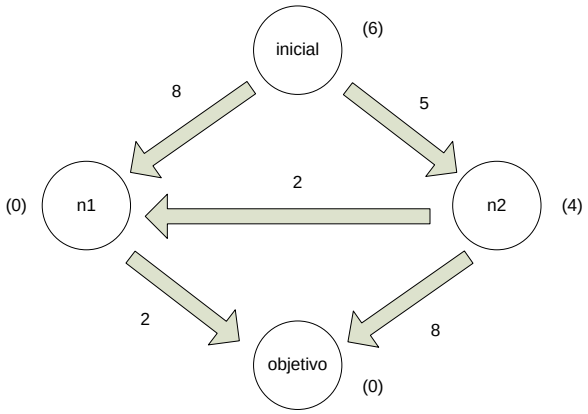
Ejercicio 1. Se trata de un ejercicio de propiedades formales de A* en un problema abstracto

Consideremos el espacio de búsqueda representado en el siguiente grafo. Los números entre paréntesis representan los valores de un heurístico h1.



Se pide responder de forma razonada, y breve, a las siguientes cuestiones. En las preguntas 1, 2, 3, 4, 6 y 7, hay que contestar utilizando las propiedades formales de A*, sin necesidad de hacer una traza de la ejecución del algoritmo. Sin embargo, en las preguntas 5, 8 y 9 hay que hacer una traza de la ejecución de A*. En estos casos hay que indicar claramente: i) la secuencia de nodos que se expanden, ii) los valores de f(n) para estos nodos, iii) el contenido de ABIERTA antes de cada iteración y iv) el camino registrado en la TABLA-A desde el inicial a cada nodo antes de cada iteración.

1.- ¿Está el heurístico h1 bien definido?

RESP: Sí está bien definido ya que $h(n) \geq 0$, $\forall n$, y $h(objetivo)=0$

2.- ¿Cuál es el valor de C* y el de h*(n), para todos los nodos del espacio de búsqueda?

RESP: $C^* = 9$, $h^*(inicial)=9$, $h^*(n1)=2$, $h^*(n2)=4$, $h^*(objetivo)=0$

3.- Si se aplica A* con el heurístico h1, ¿Encontrará la mejor solución?

RESP: h1 es admisible ya que $h1(n) \leq h^*(n)$, $\forall n$. Luego A*, luego encontrará la solución óptima.

4.- ¿Es monótono el heurístico h1?

RESP: h1 no es monótono (ni por lo tanto tampoco es consistente) ya que $4 = h1(n2) > h1(n1)+c(n2,n1) = 0+2$

5.- Supongamos ahora que se aplica un algoritmo A* al espacio anterior utilizando el heurístico h1. Además de los puntos i,ii,iii y iv anteriores, hay que indicar claramente las rectificaciones que se producen (si las hay).

RESP: La secuencia de nodos expandidos es:

1. inicial: sucesores n1 y n2

frontier =(n1 [padre: inicial, pathCost: 8] f(n1) = 8

n2 [padre: inicial, pathCost: 5] f(n2) = 9),

2. n1: sucesor obj

frontier =(obj [padre: n1 pathCost: 10] f(obj) = 10

n2 [padre: inicial, pathCost: 5] f(n2) = 9),

3. n2: sucesores n1' y obj', n1' se reinserta en frontier y obj' se descarta porque tiene un pathCost = 13 mayor que obj

frontier =(obj [padre: n1 pathCost: 10] f(obj) = 10

n1' [padre: n2, pathCost: 7] f(n1) = 7),

4. n1' sucesor obj''

frontier =(obj [padre: n1 pathCost: 10] f(obj) = 10

obj'' [padre: n1', pathCost: 9] f(obj'') = 9),

5. obj'': La solución encontrada es (inicial→n2→n1'→obj'') de coste 9. Es óptima tal y como corresponde a un heurístico admisible. Pero el estado n1 se reexpandió por el hecho de que el heurístico no es consistente.

6.- En el caso de que el heurístico h1 anterior no fuese monótono, modificar el valor del heurístico para el nodo n1, de forma que el heurístico resultante, h2, sí sea monótono.

RESP: h2(n1)=2, y h2(n)=h1(n), para el resto de los nodos. Así h2 es monótono

7.- ¿Qué solución (secuencia de nodos) encontrará A* si se aplica este nuevo heurístico h2 y cuáles serán los valores de g(n) de estos nodos en el momento de ser expandidos?

RESP: Como h2 es monótono, también es admisible, entonces encontrará la solución óptima, que en este ejemplo es única (inicial→n2→n1→objetivo). Además, por ser h2 monótono, cada vez que expanda un nodo n, g(n)=g*(n). Luego los valores de g de los nodos de este camino en el momento de ser expandidos serán g(inicial)=0, g(n2)=5, g(n1)=7, g(objetivo)=9.

8.- Hacer la traza de A* con el heurístico h2.

RESP: La secuencia de nodos expandidos es:

1. inicial: sucesores N1 y N2

frontier =(N1 [padre: inicial, pathCost: 8] f(N1) = 10

N2 [padre: inicial, pathCost: 5] f(N2) = 9),

2. N2: sucesores N1' y OBJ (el estado n1 aparece duplicado en frontier)

frontier =(N1 [padre: inicial, pathCost: 8] f(N1) = 10

OBJ [padre: N2 pathCost: 13] f(OBJ) = 13

N1' [padre: N2, pathCost: 7] f(N1') = 9),

3. N1': sucesores OBJ' (el objetivo aparece duplicado en frontier)

frontier =(N1 [padre: inicial, pathCost: 8] f(N1) = 10

OBJ [padre: N2 pathCost: 13] f(OBJ) = 13

OBJ' [padre: N1' pathCost: 9] f(obj) = 9),

4. OBJ': La solución encontrada es (inicial→N2→N1'→OBJ') de coste 9. Es óptima tal y como corresponde a un heurístico admisible. No se reinsertan nodos expandidos en frontier, como corresponde a un heurístico consistente, pero sí aparecen nodos con estados duplicados en frontier.

9.- Definir ahora otro heurístico, h3, que haga que A* encuentre la solución (inicial→n1→objetivo), si se aplica a este espacio de búsqueda. Hacer la traza correspondiente.

RESP: Para obtener esa solución, que no es óptima, necesitamos un heurístico no admisible. Si definimos el heurístico h3 de modo que h3(n1)=0 y h3(n2)=10, y h3(n)=h1(n) para el resto de los nodos tenemos un heurístico no admisible que nos lleva a la solución buscada. Ojo, puede haber heurísticos no admisibles que también nos den la solución óptima. La secuencia de nodos expandidos es:

La secuencia de nodos expandidos es:

1. inicial: sucesores N1 y N2

frontier = (N1 [padre: inicial, pathCost: 8] f(N1) = 8

N2 [padre: inicial, pathCost: 5] f(N2) = 15),

2. N1: sucesore OBJ

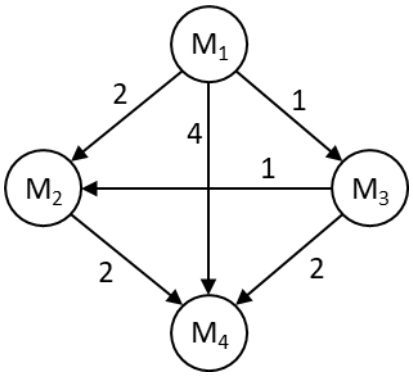
frontier = (OBJ [padre: N1 pathCost: 10] f(OBJ) = 10

N2 [padre: inicial, pathCost: 5] f(N2) = 15),

3. OBJ: La solución encontrada es (inicial→N1→OBJ) de coste 10. Solo se expanden tres estados, pero la solución no es óptima, lo cual es coherente con la falta de admisibilidad del heurístico.

Ejercicio 2. Otro de búsqueda en espacios abstractos

Consideremos el siguiente espacio de búsqueda abstracto, el algoritmo A* y los heurísticos cuyos valores se indican por extensión en la tabla



(inic.)

(obj.)

	h_1	h_2	h_3	h*
M_1	2	2	2	
M_2	0	1	2	
M_3	2	2	2	
M_4	0	0	0	

Como siempre, se pide responder de forma razonada a las siguientes cuestiones

a) ¿Cuál es el valor de C*, y el de h* para cada uno de los nodos?

RESP: C*=3, h*(M1)=3, , h*(M2)=2 , h*(M3)=2, h*(M4)=0.

b) ¿Qué propiedades tienen los heurísticos h1, h2 y h3? ¿Se puede establecer algún tipo de comparación entre ellos?

RESP: A la vista de los valores de la tabla y de los valores de h*, es evidente que se tiene la siguiente relación: Para todo n, h1(n)<=h2(n)<=h3(n)<=h*(n), pero en ningún caso se da la relación con la desigualdad estricta.

Con lo anterior, está claro que los tres heurísticos son admisibles. h1 no es monótono, ya que para M3 y M2 no se cumple la condición $h1(M3) \leq h1(M2)+c(M3,M2)$. Sin embargo, h2 y h3 sí son monótonos ya que para estos dos heurísticos se cumple la condición $h(n1) \leq h(n2)+c(n1,n2)$ para todos los nodos n1 y n2 que están conectado por una regla.

En consecuencia, la única relación formal que se puede establecer es la de "dominancia amplia" entre h2 y h3, concretamente h3 domina ampliamente a h2, que tiene como consecuencia que si h3 expande un nodo y h2 no lo expande (en el caso de que se utilicen los dos para guiar al algoritmo A*) entonces ese nodo debe cumplir que h3(n)=h2(n)=C*-g*(n). El único nodo que cumple esta condición es M4 (pero M4 es el único objetivo, con lo cual cualquier se elegirá para expandir con cualquier heurístico).

c) De acuerdo con las propiedades anteriores, ¿Qué se puede decir con respecto a la expansión del nodo M2 con cada uno de los tres heurísticos?

RESP: Dado que el heurístico h1 es admisible pero no monótono, las condiciones necesaria y suficiente de expansión de un nodo n se expresan en función de que exista

o no un camino desde el inicial a n que sea C*-acotado (necesaria) estrictamente C*-acotado (suficiente).

La primera condición se enuncia como:

$$\forall n' \in P_{inic-n}\ g_P(n') + h(n') \leq C^*$$

La segunda es igual, pero con el menor estricto. gP(n') es el coste del nodo inicial a n' a través de P.

Desde el inicial (M1) a M2 hay dos caminos, el primero P1 es M1 -2->M2, para los nodos M1 y M2 tenemos que:

gP1(M1)+h1(M1)=0+2 < C*=3

gp1(M2)+h1(M2)=2+0 < C*=3

Luego P1 es estrictamente C*-acotado y en consecuencia el nodo M2 cumple la condición suficiente de expansión y por lo tanto si ejecutamos el algoritmo A* con el heurístico h1 sobre el espacio de búsqueda anterior, el nodo M2 se expandirá.

Como los heurísticos h2 y h3 son monótonos, las condiciones de expansión de un nodo n se expresan simplemente como: g*(n)+h(n) □ C* (suficiente) y g*(n)+h(n) < C* (necesaria). Así, tenemos que:

g*(M2)+h2(M2) = 2+1 = 3 = C*

g*(M2)+h3(M2) = 2+2 = 4 > C*=3

En consecuencia, con h3 no se expandirá el nodo M3, ya que no se cumple la condición suficiente de expansión, y con h2 puede expandirse o no, ya que se cumple la condición necesaria, pero no la suficiente. Esto es coherente con el hecho de que los heurísticos mejor informados expanden menos nodos en los experimentos.

Ejercicio 3. Sobre una variante del problema del 8-puzzle

Se trata de resolver, mediante búsqueda en espacios de estados, con A* por supuesto, una variante del problema del 8-puzzle en la que en lugar de tener 8 fichas distintas (1,2,3,4,5,6,7,8), tenemos 8 fichas tales que son iguales dos a dos (1,1,2,2,3,3,4,4). El estado inicial puede ser cualquier distribución de las fichas en el tablero, por ejemplo:

$$\begin{bmatrix} 3 & 3 & 4 \\ 4 & & 1 \\ 2 & 1 & 2 \end{bmatrix}$$

Y se trata de llegar a una configuración en la que la fichas estén ordenadas de la forma siguiente:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

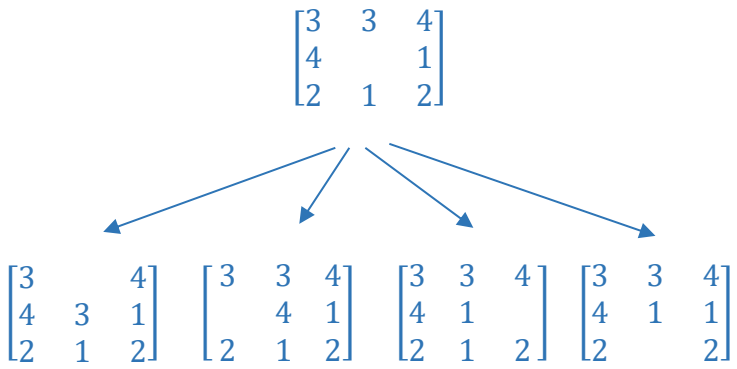
Los movimientos permitidos son los mismos que en el 8-puzzle normal, es decir se puede mover una ficha de su posición a la posición vacía si las dos posiciones son adyacentes ortogonalmente. El objetivo es también calcular la secuencia mínima de movimientos para llegar de la configuración inicial al objetivo.

Lo que se pide en este ejercicio es modelar el espacio de búsqueda y definir buenos heurísticos, **concretamente se pide responder de forma razonada a las siguientes cuestiones**:

a) ¿Cómo son los estados, las reglas y sus costes? Dibujar la parte del estado de búsqueda correspondiente al estado inicial anterior y sus sucesores.

RESP: Esto es como en el 8-puzzle convencional: Los estados son configuraciones válidas del tablero, por lo tanto, se pueden representar mediante permutaciones de las 8 fichas y un 0 para representar a la posición vacía. Por ejemplo, los estados anteriores estarían representados por las permutaciones (3 3 4 4 0 1 2 1 2) y (1 2 3 4 0 1 2 3 4) respectivamente. Las reglas aplicables a un estado vienen dadas por todas las posibilidades de mover una ficha a la posición vacía, si esta posición es adyacente ortogonalmente a la de la ficha; y los costes de todas las reglas son 1 dado que lo que se pretende en minimizar el número de movimientos.

Con todo esto, los sucesores del estado inicial anterior son los siguientes:



b) Dar una cota superior, lo más ajustada posible, del número de estados del espacio de búsqueda.

RESP: En principio el número de estados diferentes es el número de permutaciones diferentes que se pueden generar con las 8 fichas y el 0, es decir 9!. Pero en este caso como las 8 fichas son iguales 2 a 2, tendremos que dividir este número 4 veces entre 2!. Es decir la cota que podemos dar es **9!/(2*2*2*2) = 362880/16 = 22680** estados diferentes. Dado un estado inicial cualquiera, estos son en principio los estados que serían alcanzables desde el estado inicial. Un estudio más detallado nos permitiría, posiblemente, reducir este número de forma similar a cómo en el 8-puzzle normal se reduce a 9!/2, pero esto va más allá de lo que hay que hacer en esta pregunta.

c) Consideremos el heurístico $h_a(n)$ definido de forma que calcula para cada ficha la distancia ortogonal desde la posición de la ficha en el estado n a la posición más cercana en la que podría estar esa ficha en el objetivo, y luego suma todas estas distancias. Indicar el valor de este heurístico para todos los nodos dibujados en a). ¿Está h_a bien definido? ¿Es h_a admisible? ¿Es h_a monótono?

RESP: El valor del heurístico h_a para los 5 nodos de la figura anterior es (recorriendo las fichas por filas y los nodos sucesores de izquierda a derecha):

$h_a(\text{inicial}) = 2+1+1+0+1+1+1+1 = 8.$

$h_a(n_1) = 2+2+1+0+1+1+1+1 = 9.$

$h_a(n_2) = 2+1+1+1+1+1+1+1 = 9.$

$h_a(n_3) = 2+1+1+0+2+1+1+1 = 9.$

$h_a(n_4) = 2+1+1+0+1+1+2+1 = 9.$

El heurístico $h_a(.)$ está bien definido ya que $h_a(n) \geq 0$ para todo nodo n, y $h_a(\text{objetivo}) = 0$.

También es claro que es admisible, es decir $h_a(n) \leq h^*(n)$ para todo nodo n, dado que para llevar una ficha a su posición en el objetivo como mínimo hay que moverla tantas veces como su distancia ortogonal a la posición más cercana a la posición actual que puede ocupar esa ficha en el objetivo.

Para demostrar que es monótono (o consistente que es equivalente), es decir que $h_a(n_1) \leq h_a(n_2) + c(n_1,n_2)$ para todo par de nodos n_1,n_2, basta con tener en cuenta que el cambio en el valor del heurístico al pasar de un estado a un sucesor es +1, 0 ó -1, y que $c(n_1,n_2)$ es 1 si n_2 es sucesor de n_1 e ∞ en otro caso.

d) Consideremos ahora la siguiente relajación del problema: eliminamos la restricción de que no puede haber dos fichas en la misma posición, es decir se puede mover una ficha a una posición adyacente ortogonalmente aunque esta posición no esté vacía. ¿Cómo es el heurístico h_b resultante de esta relajación? Dar el valor de h_b para el estado inicial. Pista: puede ser de utilidad definir funciones: $D_{11}(n,i)$ = distancia ortogonal de la primera aparición de la ficha i en el estado n (siguiendo el orden de las filas) hasta la posición de la primera aparición de i en el objetivo. Análogamente D_{12}, D_{21} y D_{22}.

RESP: Si eliminamos la restricción de que en una casilla solamente puede haber una ficha, entonces para resolver de forma óptima el problema podríamos mover las fichas siguiendo un camino ortogonal desde su posición actual hasta una de las posiciones que cada ficha puede ocupar en el objetivo. Pero para cada par de fichas iguales, tenemos que tener en cuenta que no pueden ir las dos **finalmente** a la misma posición, por eso hay que considerar para cada par de fichas dos opciones y quedarnos con la que produzca un menor coste. El coste de resolver este problema relajado para un nodo n (y por lo tanto el valor del heurístico h_b) se puede calcular con ayuda de las funciones anteriores como:

$$h_b(n) = \sum_{i=1}^{4} min\big(D_{11}(n,i) + D_{22}(n,i), D_{12}(n,i) + D_{21}(n,i)\big)$$

El valor de este heurístico para los estados anteriores es (considerando el orden de las fichas 1, 2, 3, 4):

h_b(inicial) = min(1+3, 3+1) + min(1+3, 3+1) + min(2+1, 2+3) + min(1+0, 3+2) = 12. $h_b(n_1)$ = min(1+3, 3+1) + min(1+3, 3+1) + min(2+2, 2+2) + min(1+0, 3+2) = 13.

$h_b(n_2)$ = min(1+3, 3+1) + min(1+3, 3+1) + min(2+1, 2+3) + min(1+1, 3+1) = 13.

$h_b(n_3)$ = min(2+1, 3+2) + min(1+3, 3+1) + min(2+1, 2+3) + min(1+0, 3+2) = 11.

$h_b(n_4)$ = min(2+1, 3+2) + min(1+3, 3+1) + min(2+1, 2+3) + min(1+0, 3+2) = 11.

e) ¿Es h_b consistente? ¿Se puede afirmar que h_b domina ampliamente a h_a?

RESP: El heurístico h_b es consistente dado que se ha definido mediante el método de la relajación del problema (y ocurre que todos los heurísticos definidos mediante este método son consistentes). Además, ocurre que para todo nodo n $h_b(n) \geq h_a(n)$; esto es claro dado que para resolver el problema relajado que da lugar al heurístico h_b no se pueden mover las dos fichas iguales a la misma posición, y la suma de distancias ortogonales de cada par de fichas iguales a la posición más cercana en el objetivo al que puede ir es menor que la suma de los movimientos elementales que tendríamos que hacer para llevar las dos fichas a sus posiciones en el objetivo de la forma menos costosa posible. Por lo tanto h_b domina ampliamente a h_a.

Ejercicio 4. Cuestiones sobre el 8-puzzle, A* y heurísticos

Consideremos un heurístico h(n) tal que para cada estado suma 1 por cada ficha que no está en la columna que le corresponde en el objetivo y suma 1 también por cada ficha que no está en su fila en el objetivo.

a) Indicar de forma razonada si h está bien definido y cuáles son sus propiedades.

RESP: Está bien definido ya que si todas las fichas de n están colocadas también están en sus filas y columnas, y por lo tanto h(n)=0. Si alguna ficha está descolocada, entonces no está en su fila o no está en su columna, luego h(n)>0. Luego h(n)>=0 para todos los estados, y en particular h(objetivo)=0.

Además, h es admisible ya que para llevar una ficha que no está en su fila (columna) a la posición que le corresponde en el objetivo hay que moverla al menos una vez; y si no está ni en su fila ni en su columna habrá que moverla al menos 2 veces.

También es consistente (o monótono que en la práctica es lo mismo), monótono ya que al pasar de n1 a n2 con un movimiento de una ficha, tendremos que:

- h(n2) = h(n1)+1, si se hace un movimiento horizontal (vertical) y la ficha estaba en su columna (fila) y por lo tanto en n2 la ficha está fuera de su columna (fila)
- h(n2) = h(n1) si la ficha estaba fuera de su fila (columna) y en n2 sigue estándolo
- h(n2) = h(n1)-1 si la ficha estaba fuera de su fila (columna) y en n2 pasa a estar en su fila (columna).

b) Comparar el heurístico h con los clásicos h1 y h2 en términos de calidad de la solución y nodos expandidos.

RESP: Como los tres heurísticos son admisibles, todos encontrarán la solución óptima. Además, se cumple que, para todo n, h1(n)<=h(n)<=h2(n). En el primer caso es claro ya que cada ficha que no está en su fila o columna, tampoco está en su posición en el objetivo. En el segundo, si una ficha no está en su fila, o no está en su columna, entonces está al menos a distancia ortogonal 1 de su posición en el objetivo, y si no está ni en su fila ni en su columna, entonces está al menos a distancia ortogonal 2.

Como los tres heurísticos son monótonos, por el hecho de tener h1(n)<=h(n)<=h2(n), se cumple que h domina ampliamente a h1, y h2 domina ampliamente a h. En consecuencia, es muy poco probable que un nodo expandido por h1 sea expandido por h. Solo podría darse el caso en nodos que cumplan h1(n)=h(n)=C*-g*(n). Obviamente, la cantidad de nodos que cumplen estas condiciones son muy pocos en términos relativos. Análogamente, si consideramos h2 y h.

De modo que lo que cabe esperar, para problemas no triviales, es que h expanda menos nodos que h1 pero más que h2 para llegar a una solución.

c) Indicar los valores de h, h1 y h2 para el siguiente estado y sus sucesores.

$$\begin{bmatrix} 7 & 3 & 1 \\ 6 & & 4 \\ 2 & 8 & 5 \end{bmatrix}$$

RESP:

$$\begin{bmatrix} 7 & 3 & 1 \\ 6 & & 4 \\ 2 & 8 & 5 \end{bmatrix} \begin{matrix} h1 = 6 \\ h = 9 \\ h2 = 12 \end{matrix}$$

$$\begin{bmatrix} 7 & & 1 \\ 6 & 3 & 4 \\ 2 & 8 & 5 \end{bmatrix} \begin{matrix} h1 = 6 \\ h = 10 \\ h2 = 13 \end{matrix} \quad \begin{bmatrix} 7 & 3 & 1 \\ & 6 & 4 \\ 2 & 8 & 5 \end{bmatrix} \begin{matrix} h1 = 6 \\ h = 8 \\ h2 = 11 \end{matrix} \quad \begin{bmatrix} 7 & 3 & 1 \\ 6 & 4 & \\ 2 & 8 & 5 \end{bmatrix} \begin{matrix} h1 = 7 \\ h = 10 \\ h2 = 13 \end{matrix} \quad \begin{bmatrix} 7 & 3 & 1 \\ 6 & 8 & 4 \\ 2 & & 5 \end{bmatrix} \begin{matrix} h1 = 6 \\ h = 8 \\ h2 = 11 \end{matrix}$$

Ejercicio 5. El problema de asignación simple con búsqueda heurística

Se trata de resolver con un algoritmo A* el siguiente problema: dada una matriz cuadrada de dimensión N×N de números no negativos, calcular un subconjunto de N números tal que la suma sea mínima y de forma que no haya más de un número de una misma fila ni de una misma columna.

Si consideramos la matriz siguiente

$$\begin{pmatrix} 1 & 2 & 3 \\ 5 & 4 & 6 \\ 7 & 9 & 8 \end{pmatrix}$$

la solución óptima viene dada por el subconjunto {1, 4, 8}.

Concretamente se pide lo siguiente:

a) Dar una definición precisa del espacio de búsqueda para un valor N. Esto incluye una descripción del estado inicial, la forma de calcular sucesores para cada estado con los costes de los operadores correspondientes y una descripción de los objetivos. Obviamente el espacio debe ser completo y a la vez lo más reducido posible.

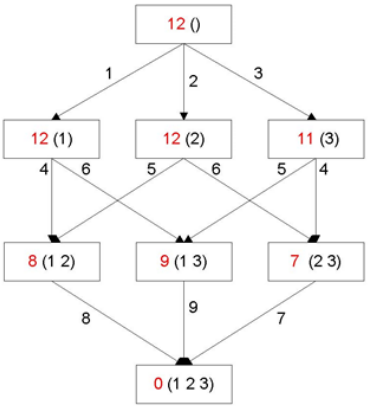
Dibujar el espacio de búsqueda completo para el ejemplo dado por la matriz anterior.

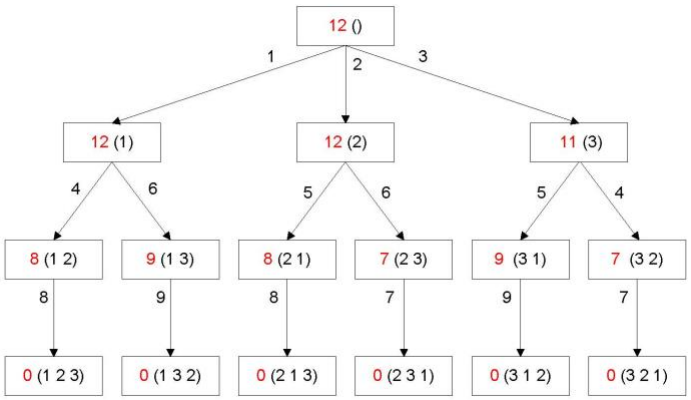
RESP: Un estado se puede definir mediante un subconjunto de elementos de {1..N}. La interpretación de uno de estos estados puede ser la siguiente: si el tamaño del subconjunto es L, entonces para llegar a este estado ya hemos elegido números de las filas 1..L y por lo tanto para llegar a una solución válida tendremos que elegir números de las filas L+1..N tales que no estén en las columnas 1..L y que dos números no estén en la misma fila ni en la misma columna. Así, un estado representa un problema como el original pero restringido a una matriz de tamaño (N-L)×(N-L). Con esta interpretación, el estado inicial vendrá dado por el conjunto vacío y el único estado final vendrá dado por el conjunto total {1,2,..,N}. Dado un estado intermedio n de tamaño L tendrá N-L sucesores, cada uno de los cuales se genera añadiendo al conjunto de n uno de los elementos de {1,..,N} que no están en n. El coste del operador correspondiente es el valor del elemento de la matriz en la posición (L+1,k) si k es el elemento añadido al conjunto.

El espacio definido de esta forma tiene estructura de grafo.

Se puede optar por una definición alternativa en la que el espacio de búsqueda es un árbol. Esta se ilustra con un ejemplo en el apartado b). En este caso los nodos son listas ordenadas de números que indican los índices de las columnas elegidas en cada paso, por ejemplo el estado (2 3) representa que hemos elegido el segundo elemento de la primera fila y el tercero de la segunda fila. Entre las dos opciones, es esperable que la primera sea la mejor ya que resulta en un espacio de búsqueda mucho más reducido.

Las dos figuras siguientes muestran los espacios de búsqueda generados a partir de la matriz anterior de tamaño 3×3:





b) [1 punto] Definir un heurístico h1 "lo mejor posible" para el problema y justificar qué propiedades tiene.

Indicar los valores que toma el heurístico en cada uno de los nodos del espacio pintado en a).

RESP: El problema asociado a cada estado se puede relajar si eliminamos la restricción de que cada número elegido deba estar en una columna diferente (manteniendo que debe estar en columnas distintas a las de los índices que definen el propio estado). De esta forma, el problema relajado se resuelve de forma óptima sin más que elegir el número menor de cada una de las filas L+1..N que no esté en ninguna de las columnas cuyos índices definen el estado. Dado un estado n, h1(n) lo definimos como el coste óptimo del problema relajado correspondiente al nodo n.

Dado que el heurístico h1 se obtiene mediante el método de la relajación del problema, es consistente.

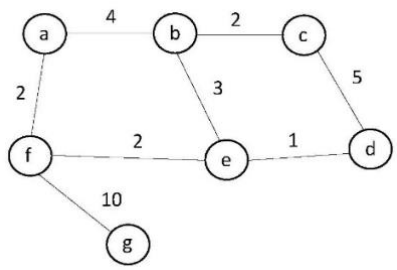
Las figuras anteriores muestran estos valores para cada uno de los nodos de los dos espacios de búsqueda (árbol y grafo). Son los números en rojo.

c) [1 punto] Indicar los nodos que no se expandirán con el heurístico h anterior y con el heurístico h0(n)=0 para todo nodo n.

RESP: Dado que se trata de un heurístico monótono, la condición que debe cumplir un nodo para que no se expanda es lo contrario de la condición necesaria de expansión, es decir: f(n) = g(n)+h(n) > C*. En este caso C* = 13. Con el heurístico h1 en el grafo solo hay 4 nodos que no cumplen esta condición, los cuatro que forman parte de la solución óptima. Con el heurístico h0 solamente la cumplen los 5 nodos solución que no representan soluciones óptimas.

Ejercicio 6. El problema del vigilante con búsqueda en espacios de estados

Tenemos un grafo que representa conexiones entre ciudades, como por ejemplo el siguiente:



Una conexión entre dos ciudades expresa que se puede ir de una a la otra con el coste que indica el arco, pero también que se puede ver una ciudad desde la otra (esto sin coste alguno). Si dos ciudades no están conectadas con un arco, no se puede ir de una a la otra directamente, ni se puede ver una desde la otra. Una ciudad se ve desde sí misma, por supuesto.

El objetivo es calcular una ruta que parta de una ciudad concreta, en el ejemplo anterior es la ciudad a, que pase por una serie de ciudades, pero solamente una vez por cada una de ellas (y no tiene que pasar por todas las ciudades del grafo) y que termine de nuevo en la ciudad de partida, de forma que todas las ciudades del grafo puedan ser vistas desde alguna de las ciudades de la ruta. Además, la ruta debe ser de coste mínimo.

En el ejemplo anterior hay varias soluciones, la óptima es la ruta (a,b,e,f,a) con coste 11. En este problema es posible que no exista solución alguna, dependiendo de cómo sean las conexiones del grafo.

En este ejercicio se pide lo siguiente:

1.- Indicar otra solución al problema del grafo anterior que sea distinta a (a,b,e,f,a) y su coste. Indicar también un ciclo que empiece y termine en la ciudad a y que no sea solución del problema. Dar también un grafo de conexiones distinto al anterior para el que el problema no tenga solución.

RESP: La ruta (a,b,c,d,e,f,a) es otra solución del problema con coste 16, por lo tanto no óptima. La ruta (a,b,a) no es solución del problema ya que no permite ver a las ciudades d, c y g. El grafo a – b – d – e no tiene solución ya que después del recorrido (a,b,d), aunque todas las ciudades están vistas, no se puede regresar a la ciudad a a través de ciudades no visitadas.

2.- Modelar el problema para resolverlo con búsqueda en espacios de estados. Es decir, definir el espacio de búsqueda (estados, reglas, costes, estado inicial, estados objetivos). Dibujar una parte significativa del espacio de búsqueda para la instancia del problema definida por el grafo anterior.

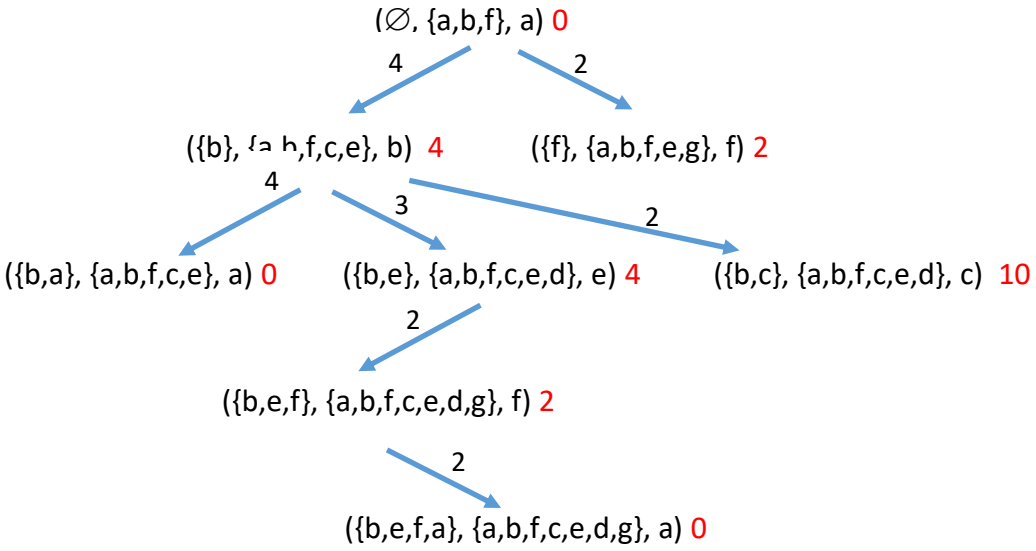
RESP: En cada estado hay que tener registradas las ciudades que ya han sido vistas y las que ya sido visitadas, así como la ciudad actual. Con esta representación, el subproblema que representa un estado es: ir desde la ciudad actual a la ciudad de partida pasando por ciudades no visitadas, no más de una vez por cada una, de forma que se puedan ver todas las ciudades no vistas, y todo ello con el menor coste posible.

Así un estado será una tripleta de la forma (Visitadas, Vistas, Actual). Utilizando la idea de la pista (a), el estado inicial será

(∅, Conjunto de ciudades que se ven desde la ciudad de partida, ciudad de partida)

Es decir, inicialmente no registramos en el conjunto de ciudades visitadas a la ciudad de partida. Un estado es objetivo si estamos en la ciudad de partida y el conjunto Vistas incluye a todas las ciudades. Los sucesores de un estado vienen dados por las posibilidades de ir directamente desde la ciudad actual a una ciudad no visitada, el coste de cada regla es el de la conexión correspondiente y el nuevo estado es como el anterior cambiando la ciudad actual, añadiendo ésta a las Visitadas y añadiendo también las que se ven desde ésta a las Vistas.

Para el problema representado por el grafo de conexiones anterior, el espacio de búsqueda es el siguiente (una parte representativa)



3.- Proponer un heurístico para resolver el problema con el algoritmo A*, e indicar sus propiedades. Dar los valores de este heurístico para un conjunto significativo de los nodos del espacio anterior.

RESP: Se puede relajar la restricción de que "hay que ver las ciudades no vistas en el estado actual" a través de la ruta desde la ciudad actual a la ciudad de partida. En ese caso, la solución óptima del problema relajado es el camino más corto desde la ciudad actual a la de partida que no pase por las ciudades ya visitadas. Ese camino se calcula mediante el algoritmo de Dijsktra aplicado al "grafo residual" (el grafo inicial sin las ciudades visitadas). Para el espacio de búsqueda anterior, se indican los valores de este heurístico al lado de cada nodo, en rojo.

El heurístico es consistente por haber sido diseñado utilizando el método de la relajación del problema.

PISTAS: (a) No hay que confundir las ciudades visitadas con las ciudades vistas. Cuando llegamos a una ciudad desde otra, la registramos como visitada. Cuando estamos en una ciudad vemos las adyacentes. (b) El algoritmo de Dijskstra puede ser de utilidad.

Ejercicio 7. Planificación de tareas (scheduling) con búsqueda en espacios de estados.

Esta vez consideramos el problema de planificación de un conjunto de n trabajos $\{J_1,\ldots,J_n\}$ en m máquinas $\{M_1,\ldots,M_m\}$. Cada trabajo J_i tiene una duración $d_i>0$, $i=1,\ldots,n$; y la ejecución del trabajo J_i en la máquina M_j tiene un coste $C_{ij}>0$, $i=1,\ldots,n$; $j=1,\ldots,m$. Cada máquina puede ejecutar cualquier trabajo, pero solo uno a la vez.

El objetivo es planificar los n trabajos en la m máquinas de forma que su ejecución no exceda un tiempo $T_{max}>0$ y que el coste total de ejecución sea mínimo.

En primer lugar, hay que resolver el problema con búsqueda en espacios de estados, para ello se pide:

1.- Dar una interpretación precisa del espacio de búsqueda (estados, estado inicial, estados objetivo, reglas y costes de las reglas)

RESP: En principio, la solución más simple sería considerar que un estado es una situación en la que tenemos algunos trabajos planificados en cada máquina. Esto sería suficiente ya que nos permitiría saber el tiempo de disponibilidad de cada máquina y los trabajos que quedan sin planificar. El inconveniente de esta representación es que el tamaño de los estados iría creciendo con la profundidad, con lo que tendríamos un problema de espacio de memoria.

Para evitar que el tamaño de los estados crezca con la profundidad, otra opción más adecuada es considerar que un estado es una tupla de tres elementos:

$$(JR, (T_{M1},\ldots, T_{Mm}), (J,M))$$

en la que

- JR es el conjunto de trabajos que quedan sin planificar
- $(T_{M1},\ldots, T_{Mm})$ son los tiempos de ocupación de las máquinas $1,\ldots,m$ con los trabajos planificados hasta el momento, tal que $T_{Mi} \leq T_{max}$.

(J,M) indica que el último trabajo planificado es $J\in\{J_1,\ldots,J_n\}$ y está asignado a la máquina $M \in\{M_1,\ldots,M_m\}$.

Con esta representación, el estado inicial será

$$(\{J_1,\ldots,J_n\},(0,\ldots,0),(null,null))$$

Y los estados objetivo serían cualesquiera de la forma

$$(\varnothing, (T_{M1},\ldots, T_{Mm}), (J,M))$$

Los operadores, o reglas, que permiten pasar de un estado $(JR, (T_{M1},\ldots,T_{Mm}), (J,M))$ a un sucesor consisten en elegir un trabajo $J\in JR$ y asignarlo a una máquina $M\in\{M_1,\ldots,M_m\}$, y por supuesto actualizar la componente T_M de $(T_{M1},\ldots,T_{Mm})$ con la duración del trabajo J. El coste de la regla es el coste de ejecución del trabajo J en la máquina M.

Es importante mantener el par (J,M) en cada estado ya que solo así se puede recuperar la planificación completa al recorrer los nodos desde el objetivo encontrado hasta el nodo inicial.

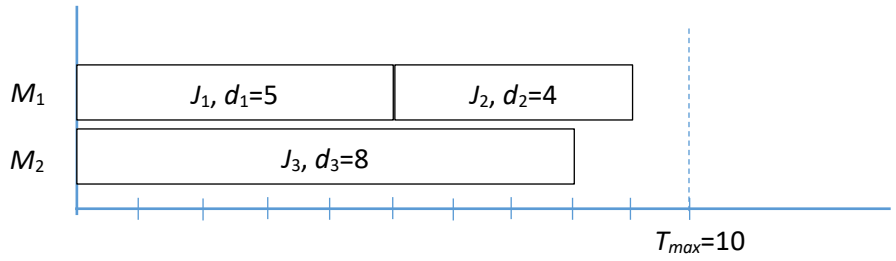
Dado que el orden de procesamiento de los trabajos en las máquinas es irrelevante para el cálculo de los costes y tiempos de procesamiento en las máquinas (los trabajos siempre se ejecutan uno a continuación del anterior sin pérdida de tiempo), se puede simplificar el espacio de búsqueda de forma que al ir aplicando las reglas desde el estado inicial los trabajos se consideran en un determinado orden, por ejemplo 1,…,*n*. Con esta idea, un estado se puede representar como una tupla con solamente dos elementos

$$((T_{M1},\ldots, T_{Mm}), (J_i,M))$$

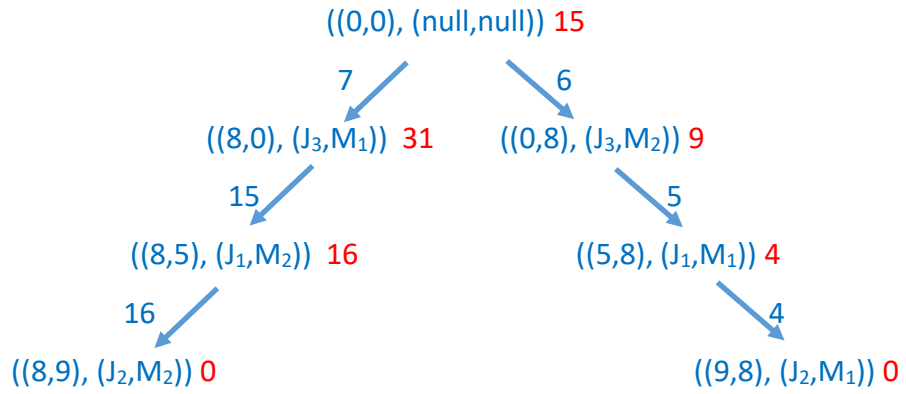
El estado inicial y los objetivos se deducen de forma simple: el inicial será ((0,…,0)(null,null)) y los objetivos serán de la forma$((T_{M1},\ldots, T_{Mm}), (J_n,M))$. Las reglas aplicables al estado $((T_{M1},\ldots, T_{Mm}), (J_i,M))$ serán tantas como máquinas cumplan $T_{Mi} \leq T_{max} + d_{i+1}$ y consistirán en planificar el trabajo J_{i+1} en cada una de estas máquinas. Si esta condición no se cumple para ninguna de las máquinas, el estado no tendrá sucesores, es decir es un callejón sin salida, ya que claramente es imposible llegar a una solución a partir de él. Teniendo en cuenta esto último, en lugar de considerar la ordenación 1,…,*n* de los trabajos, sería preferible ordenarlos por sus duraciones de mayor a menor, ya que de esta forma se alcanzarían antes los callejones sin salida y el espacio de búsqueda sería más reducido.

2.- Dibujar el espacio de búsqueda completo para la instancia del problema definida por los siguientes datos: n=3, m=2, d_1=5, d_2=4, d_3=8, C_{11}=5, C_{12}=15, C_{21}=4, C_{22}=16, C_{31}=7, C_{32}=6 y T_{max}=10 (con estos datos si T_{max}<9 el problema no tendría solución).

El gráfico de Gantt siguiente muestra una solución para esta instancia. El coste de esta solución es $C_{11} + C_{21} + C_{32} = 5 + 4 + 6 = 15$



RESP: De acuerdo con lo anterior, vamos a considerar el orden de los trabajos (J_3,J_1,J_2). Con este orden el espacio de búsqueda para esta instancia del problema es:



En este ejemplo simple ocurre que h(n)=h*(n) para los 7 nodos, pero en general esto no se cumple.

Además, consideramos que este espacio de búsqueda se recorre con el algoritmo A*, para ello se pide también:

3.- Definir un heurístico lo mejor informado posible, preferiblemente a través del método de relajación del problema. Indicar qué propiedades tiene el heurístico y dar sus valores para los nodos del espacio de búsqueda anterior.

RESP: Este problema se puede relajar de varias formas. Una opción es relajar la restricción de que las máquinas solo pueden realizar un trabajo a la vez. Otra posibilidad es suponer que no existe T_{max}, en realidad suponer que T_{max}=infinito. Vamos a considerar la primera.

Si una máquina puede realizar varios trabajos a la vez, la solución óptima del problema relajado que representa un estado $((T_{M1},\ldots,T_{Mm}),(J_i,M))$ se puede calcular en tiempo polinomial. El algoritmo consiste en asignar cada uno de los trabajos restantes $J_{i+1},\ldots,J_n$ a la máquina de $\{M_1,\ldots,M_m\}$ en la que tenga menor coste de ejecución, siempre y cuando su duración lo permita, es decir que no se exceda el límite T_{max} en esa máquina.

El coste de este heurístico para los nodos del espacio anterior es el valor que se indica en rojo para cada uno de los estados.

El heurístico es consistente como consecuencia de haber aplicado el método de relajación del problema.

Un ejercicio interesante sería considerar cómo el es heurístico que sale de la segunda de las relajaciones comentadas anteriormente. No es igual que el anterior; además, al tratarse de relajaciones distintas, no serán estrictamente comparables.

Ejercicio 8. El problema MAX_CUT con búsqueda heurística

El famoso problema MAX_CUT se puede enunciar del siguiente modo: dado un grafo no dirigido G =(V,E) con costes positivos en los arcos, se trata de calcular una partición del conjunto de nodos V en dos subconjuntos V_1 y V_2 (es decir $V = V_1 \cup V_2$ y $V_1 \cap V_2 = \emptyset$). Si denotamos por E_1 al subconjunto de arcos de E que conectan pares de nodos de V_1 (análogamente E_2), y por Coste(E_1) la suma de los costes de los arcos de E_1 (análogamente Coste(E_2)), el objetivo es encontrar la partición (V_1,V_2) que minimice el valor de Coste(E_1)+ Coste(E_2).

En este ejercicio se pide resolver el problema MAX_CUT utilizando búsqueda en espacios de estados, en particular el algoritmo A*. Más concretamente:

1.- Describir de forma precisa los estados y las reglas, es decir el espacio de búsqueda, de forma genérica, indicando si es un árbol o un grafo (tened en cuenta que puede haber reglas de coste 0 siempre y cuando no se generen ciclos en el espacio de búsqueda de coste 0, o caminos de longitud infinita y coste 0).

RESP: Dado que una solución del problema viene dada por un par de conjuntos (V_1,V_2) tales que V_1 y V_2 son disjuntos y contienen a todos los nodos del grafo, por ejemplo el par ({A,B,E} {C,D}) para el grafo del apartado siguiente, lo natural parece representar los estados por pares de subconjuntos de nodos (C_1, C_2) tales que C_1 y C_2 sean disjuntos y que su unión contenga un subconjunto de nodos del grafo. Así, el par ({A},{E,D}) sería en principio un estado válido.

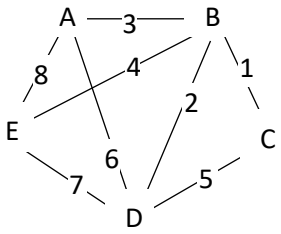
Los sucesores de un estado (C_1, C_2) pueden venir dados, en principio, por todas las opciones de añadir una ciudad no incluida en $C_1 \cup C_2$ a C_1 y a C_2. Así, el estado ({A},{E,D}) tendría, entre otros, como sucesores a los estados ({A,C},{E,D}) y ({A},{E,D,C}). El coste de cada regla sería la suma de los costes de las conexiones entre la nueva ciudad y todas las ciudades ya incluidas en C_1, si la nueva ciudad se incluye en C_1, (o en C_2 si se incluye en este conjunto). En el ejemplo anterior, los costes serían: 0 (coste entre A y C) y 5 (0 coste entre C y E, coste 5 entre C y D). El estado inicial podría ser el par ($\emptyset$,$\emptyset$). La solución óptima en este ejemplo es ({A,C,E},{B,D}) y tiene un coste C*=10.

Con estas ideas, el espacio de búsqueda sería un grafo acíclico pero con muchas conexiones ya que es evidente que en general habrá muchos caminos para llegar desde el nodo inicial a un nodo intermedio. Pero todos estos caminos tienen el mismo coste; con lo que, si fuésemos capaces de definir un espacio que solamente contenga uno de estos caminos, el espacio sería mucho más reducido y representaría las mismas soluciones, es decir sería completo.

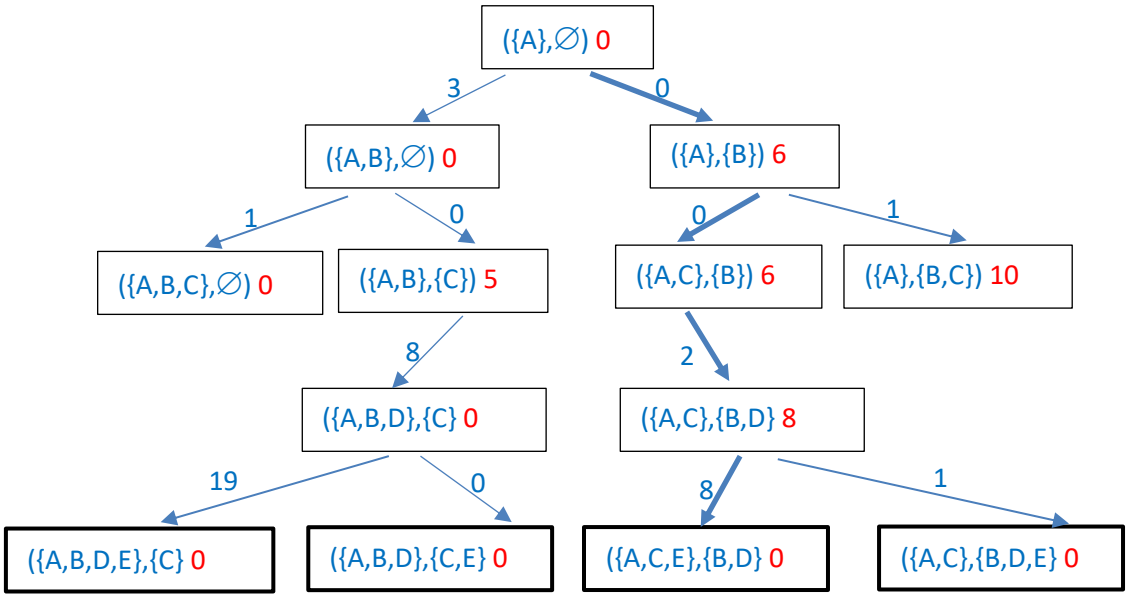
Para ello, lo que podemos hacer es establecer un orden lineal entre las ciudades (de forma similar a cómo se establece un orden de filas en el problema de las N-reinas), e ir desarrollando el espacio de búsqueda por niveles: para los sucesores del inicial se considera solo la primera ciudad, para los sucesores de estos la segunda y así sucesivamente. Así tendremos un espacio de búsqueda en forma de árbol binario de N+1 niveles, si hay N ciudades. Al igual que en las N-reinas, podemos eliminar simetrías

si partimos de un estado inicial en el que la primera ciudad está en el conjunto C_1, así nunca puede estar en C_2, y la altura del árbol se reduce en 1.

2.- Dibujar una parte representativa del espacio de búsqueda para el problema dado por el siguiente grafo



RESP: Una parte significativa del espacio de búsqueda debe contener al menos una solución y un conjunto de nodos intermedios en varios niveles del árbol. Consideramos el orden A,B,C,D,E.



3.- Diseñar un heurístico razonable, a ser posible utilizando el método de la relajación del problema. Indicar las propiedades del heurístico, y dar sus valores para la fracción del espacio de búsqueda dibujado anteriormente. PISTA: una forma de relajar el problema es prescindir de algunos de los arcos.

RESP: En este caso, dado el subproblema planteado por un estado (C_1, C_2), se puede relajar el problema prescindiendo de los arcos que unen ciudades no incluidas en $C_1 \cup C_2$. De esta forma el problema relajado se resuelve de forma óptima en tiempo polinomial sin más que recorrer las ciudades no incluidas en $C_1 \cup C_2$, y para cada ciudad c calculamos Coste(c, C_1) como la suma de los costes entre c y todas las ciudades de C_1, análogamente Coste(c, C_1). Así, en el problema relajado incluimos la ciudad c en el subconjunto con menor coste, y así obtenemos el coste óptimo del problema relajado.
El heurístico es consistente al haber sido diseñado por el método de la relajación del problema.

Los valores de este heurístico para cada uno de los nodos del espacio anterior son los números que aparecen en color rojo. Nótese que Coste(c, ∅) = 0, ya que no hay arcos entre c y las ciudades de ∅.

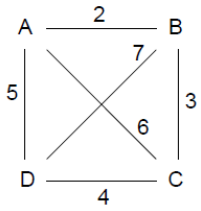
Ejemplo:

h[({A,B},{C})] = min(Coste(D,{A,B}),Coste(D,{C})) + min(Coste(E,{A,B}),Coste(E,{C})) = min(6+2,5) + min(8+4,0) = 5+0 = 5 < h*[({A,B},{C})] = 8.

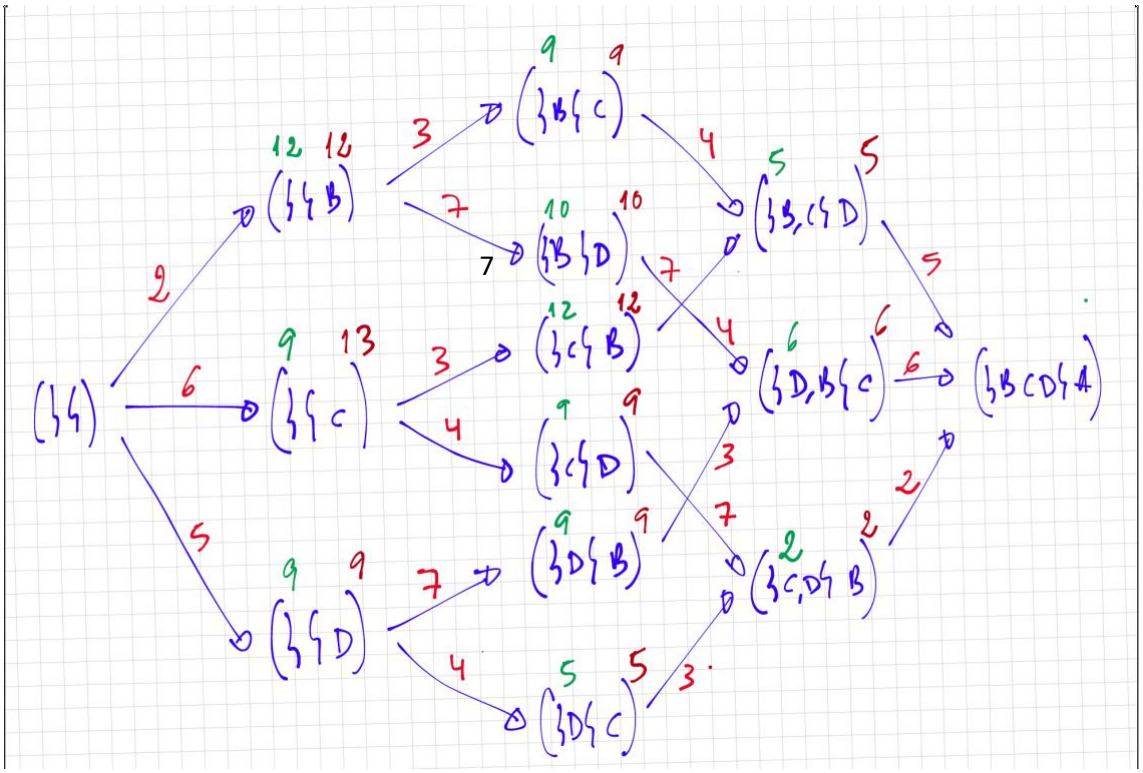
Ejercicio 9. Dos versiones del TSP con el algoritmo A*

Se trata de utilizar el algoritmo A* para resolver dos variantes del problema del viajante de comercio (TSP). La primera es la variante clásica en la que un viajante debe partir de una ciudad A, pasar por el resto de las ciudades una y solo una vez y volver a la ciudad de partida con un coste mínimo. En este caso se pide:

a) Describir de forma genérica el espacio de búsqueda para un problema con N ciudades, conectadas directamente todas con todas, y dibujar el espacio completo para la siguiente instancia del problema:



RESP: Un estado se puede representar por tres elementos: la ciudad de partida, el conjunto de ciudades por las que ya pasó el viajante y la ciudad actual. Los sucesores de un estado vienen dados por cada una de las ciudades no visitadas, y el coste para ir a cada uno de ellos desde el actual es el coste desde la ciudad actual a la nueva ciudad. Para el problema anterior, el espacio de búsqueda completo es el siguiente, los costes de los arcos se indican en color rojo:



b) Describir un heurístico, lo mejor posible, para el problema. Justificar sus propiedades y dar los valores que toma este heurístico para todos los nodos del espacio de búsqueda del ejemplo anterior.

RESP: Uno de los mejores heurísticos para este problema es el que consiste en calcular el árbol de expansión mínimo del grafo "residual", es decir el grafo de conexiones eliminando las ciudades ya visitadas, salvo la inicial y la actual, así como los arcos que conectan a las ciudades eliminadas con otras ciudades. **El arco que une la ciudad inicial con la actual se elimina también, salvo que sea el único.** El valor de este heurístico se muestra en color verde (**hay algunos errors corregidos**) en la figura anterior. En morado están los valores de h*(n) que como se puede ver son mayores o iguales que h(n) para todo nodo n.

c) Consideremos ahora la versión del TSP con "ventanas de tiempo". Es como el anterior, con una restricción añadida: el viajante tiene que pasar por cada ciudad X en un instante del intervalo $[t_{Xi},t_{Xf}]$. En este caso se pide lo mismo que para la versión anterior. Como ejemplo concreto consideraremos el problema anterior en el que las ventanas de tiempo para las ciudades B, C, y D son respectivamente [8,19], [5,7] y [9,17].

RESP: En este caso, el espacio de búsqueda se define de una forma análoga. Pero debido a la nueva restricción los estados alcanzables desde uno dado se reducen a los definidos por las ciudades no visitadas que son alcanzables desde la ciudad actual en un instante del intervalo correspondiente. En cuanto al heurístico, en principio sirve el mismo que para el TSP original. Dado que el TSP original es en realidad una relajación del TSP con ventanas de tiempo, cualquier heurístico admisible para el TSP original también lo es para el TSP con ventanas de tiempo. El espacio de búsqueda completo para el problema anterior con las ventanas de tiempo indicadas es el siguiente: