Universidad de Granada
Departamento de Ciencias de la Computación
e Inteligencia Artificial

DECSAI

Curso 2019/20

Técnicas de los Sistemas Inteligentes

Entrega de problemas - Evaluación del curso

Fecha de inicio a partir del 04-06-2020 08:00

Entrega antes del 06-06-2020 08:00

Nombre: Francisco Javier Bolívar Expósito

Problema 1.-

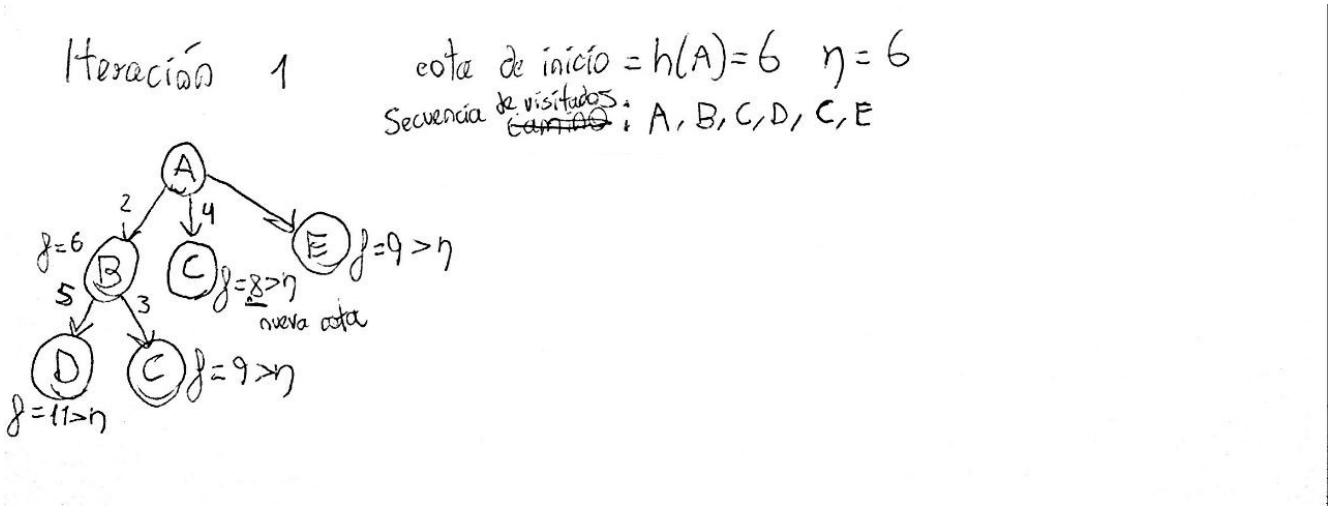
1. El número de iteraciones que han sido necesarias para alcanzar la solución
    a. Iteraciones=5
2. Para cada iteración, cota utilizada y secuencia de nodos expandidos en dicha iteración.
    a. Iteración 1
        i. Cota utilizada: 6
        ii. Secuencia de nodos: A, B, C, D, C, E
    b. Iteración 2
        i. Cota utilizada: 8
        ii. Secuencia de nodos: A, B, C, D, C, G, H, E
    c. Iteración 3
        i. Cota utilizada: 9
        ii. Secuencia de nodos: A, B, C, G, H, D, C, G, H, E, F
    d. Iteración 4
        i. Cota utilizada: 10
        ii. Secuencia de nodos: A, B, C, G, H, D, C, G, I, H, E, F
    e. Iteración 5
        i. Cota utilizada: 11
        ii. Secuencia de nodos: A, B, C, G, I, H, D, C, H, C, G, I
3. Camino solución obtenido y su coste.
    a. Camino solución: A, C, G, I
    b. Coste del camino: 11

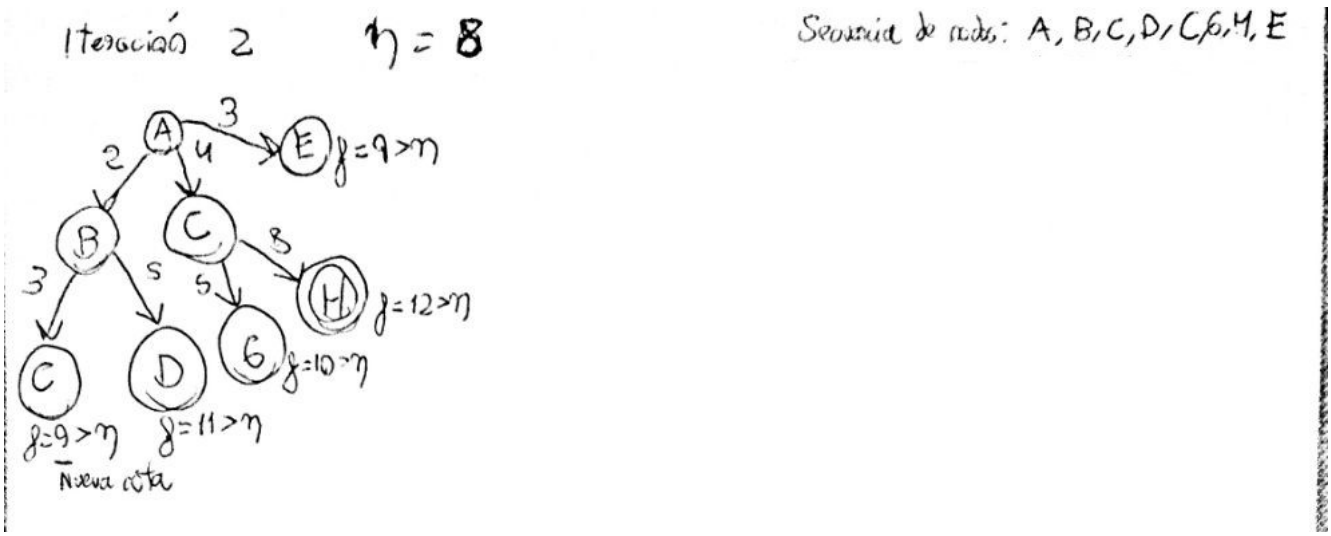
4. Aparte de los datos básicos anteriores, una descripción en texto de cómo se ha resuelto el problema, y documentos gráficos de la resolución a mano.

Usando IDA* se ha simulado a mano el proceso que seguiría el algoritmo de búsqueda. Exploramos los nodos con una búsqueda por profundidad, seleccionando los hijos en orden alfabético, pero en el momento en el que un nodo n tenga f(n) = g(n) + h(n) mayor que la cota de la iteración actual se corta esa rama y se hace backtracking volviendo a explorar los hijos de un nodo visitado anteriormente. Si no se puede explorar ningún otro camino se termina la iteración.

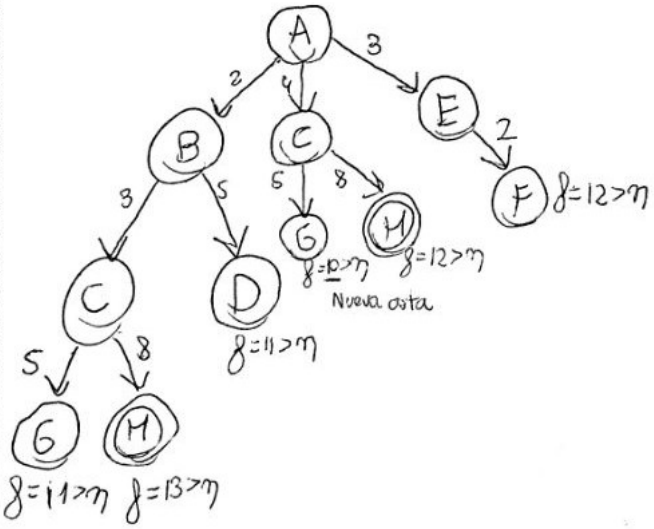
La cota se inicializa al valor heurístico h(S) del nodo inicial y aumenta en cada iteración al minimo coste total (f) de todos los nodos que han superado la cota.

Resolución a mano:



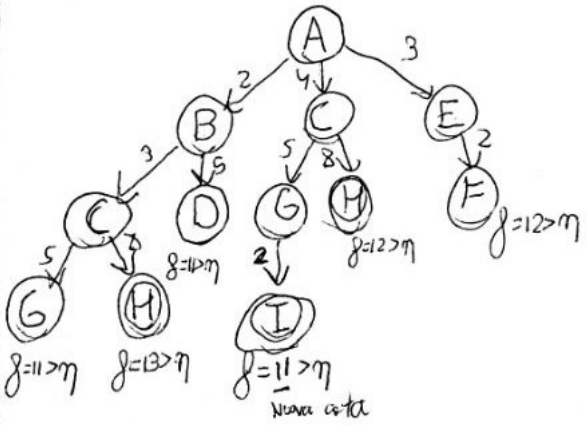


Iteración 3 $\eta = 9$



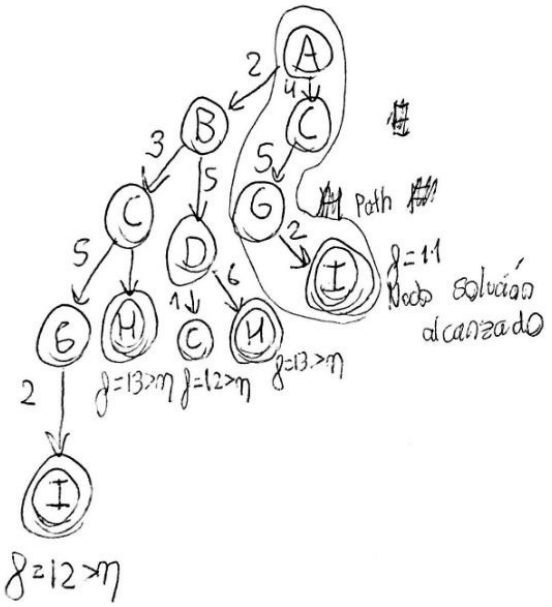
Secuencia: A, B, C, G, H, D, C, G, H, E, F

Iteración 4 $\eta = 10$



Secuencia: A, B, C, G, H, D, C, G, I, H, E, F

Iteración 5 $\eta = 11$



Secuencia: A, B, C, G, I, H, D, C, H, C, G, I

Problema 2.-

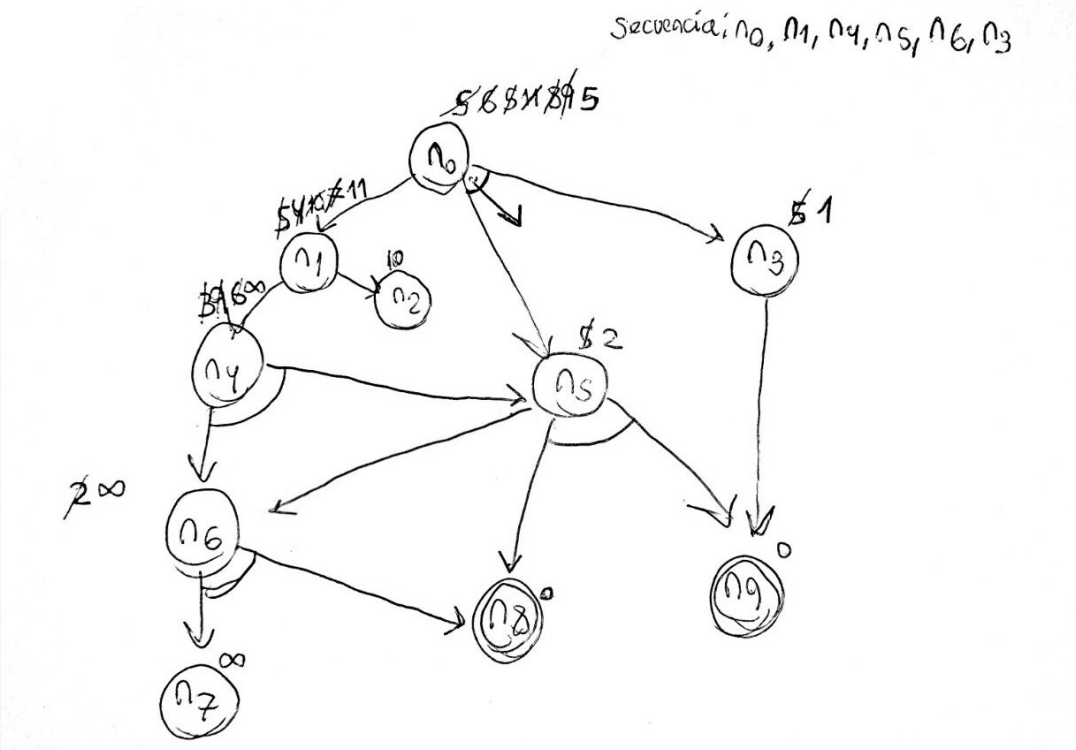
1. Orden de expansión de los nodos a lo largo del proceso: n0, n1, n4, n5, n6, n3
2. Secuencia de valores del nodo de inicio n0 a lo largo del proceso: 5, 6, 5, 11, 8, 9, 5
3. Coste de la solución y nodos que la componen.
    a. Coste de la solución: 5
    b. Nodos de la solución: n0, n5, n3, n8, n9
4. Aparte de los datos básicos anteriores, una descripción de cómo se ha resuelto el problema, y documentos gráficos de la resolución a mano.

Empezamos con un grafo que solo contiene el nodo de inicio.

Hasta encontrar el camino solución:

- Trazar el mejor camino actual desde inicio (aquel que teniendo en cuenta la longitud del camino y la longitud esperada en los nodos terminales tenga un menor valor)
- Seleccionar un nodo para continuar el mejor camino, empezando primero por los que dentro de este mejor camino tenga una mayor heurística.
- Generar los sucesores del nodo e incluirlos de forma adecuada en el grafo
- Propagar la información obtenida hacia arriba en el grafo

Resolución a mano:



Problema 3.-

1. Soluciones obtenidas.
   Se ha obtenido la conclusión de que no es posible obtener una asignación completa y consistente, por lo que no existe solución que satisfaga las restricciones.
2. Tablas con el proceso seguido incluyendo las vueltas atrás que se hayan producido.

Evolución del dominio de las variables forzando arco-consistencia. A = 1				
Arcos	dom(A)	dom(B)	dom(C)	dom(D)
A< B	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
B > A	1	2, 3, 4	2, 3, 4	1, 2, 3, 4
D < C	1	2, 3, 4	2, 3, 4	1, 2, 3
C > D	1	2, 3, 4	2, 3, 4	1, 2, 3
A = C		2, 3, 4	2, 3, 4	1, 2, 3

Como el dominio de A ha quedado vacío hacemos backtracking y la asignación realizada de A = 1 no es válida, probamos ahora con A = 2.

Evolución del dominio de las variables forzando arco-consistencia. A = 2				
Arcos	dom(A)	dom(B)	dom(C)	dom(D)
B > A	2	3, 4	1, 2, 3, 4	1, 2, 3, 4
C = A	2	3, 4	2	1, 2, 3, 4
C = B	2	3, 4		1, 2, 3, 4

Dominio de C queda vacío. Backtracking, probamos con A = 3.

Evolución del dominio de las variables forzando arco-consistencia. A = 3				
Arcos	dom(A)	dom(B)	dom(C)	dom(D)
B > A	3	4	1, 2, 3, 4	1, 2, 3, 4
C = A	3	4	3	1, 2, 3, 4
C = B	3	4		1, 2, 3, 4

Dominio de C queda vacío. Backtracking, probamos con A = 4.

Evolución del dominio de las variables forzando arco-consistencia. A = 4				
Arcos	dom(A)	dom(B)	dom(C)	dom(D)
B > A	4		1, 2, 3, 4	1, 2, 3, 4

Dominio de B vacío. Hacemos backtracking pero ya no quedan ramas a explorar, por lo que no se ha encontrado solución.

3. Aparte de los datos básicos anteriores, una descripción de cómo se ha resuelto el problema, y documentos gráficos de la resolución a mano.

    - Usamos el algoritmo de mantenimiento de Arco Consistencia (MAC). Inicializamos una cola de arcos dirigidos añadiendo por cada restricción un arco entre X e Y y un arco entre Y e X.
    - Asignamos un valor a una variable dentro de su dominio.
    - Para todas las variables no asignadas se recorre la cola, comprobando arco-consistencia y forzando a que sean arco-consistentes, para esto:
        - Para que un arco sea consistente para todo valor x_i hay algun y_i permitido. Por lo tanto comprobamos para cada valor de X si existe al menos algún valor de Y que cumpla la restricción, si no es así eliminamos este valor del dominio de X para mantener la consistencia en el dominio.
        - Si durante este proceso algún dominio se queda vacío no hay solución posible con la asignación realizada, por lo que hacemos backtracking y se repiten los pasos explorando otras asignaciones posibles.

Resolución a mano:

Var = A, B, C, D

dom = {1, 2, 3, 4}

Arcos (Necesario comprobar X arco consistente Y y Y arco consistente X)

A < B D < C A = C D < A B = C

B > A C > D C = A A > D C = B

Cola de arcos	dom(A)	dom(B)	dom(C)	dom(D)
A < B	{1}	{1,2,3,4}	{1,2,3,4}	{1,2,3,4}
B > A	{1}	{2,3,4}	{2,3,4}	{1,2,3,4}
D < C	{1}	{2,3,4}	{2,3,4}	{1,2,3}
C > D	{1}	{2,3,4}	{2,3,4}	{1,2,3}
A = C	{}	{2,3,4}	{2,3,4}	{1,2,3}
C = A				
B = C				
C = B				

A=1

• Asignamos A = 1

• Dominio vacío, realizamos backtracking y asignamos la var A a un valor diferente.

Arcos	dom(A)	dom(B)	dom(C)	dom(D)
B > A	{2}	{3,4}	{1,2,3,4}	{1,2,3,4}
C = A	{2}	{3,4}	{2}	{1,2,3,4}
C = B	{2}	{3,4}	{}	{1,2,3,4}

A=2

• Asignamos A = 2

• Dominio vacío, Backtracking A = 3

B > A	{3}	{4}	{1,2,3,4}	{1,2,3,4}
C = A	{3}	{4}	{3}	{1,2,3,4}
C = B	{3}	{4}	{}	{1,2,3,4}

A=3

• Dominio vacío, Backtracking A = 4

B > A	{4}	{}	{1,2,3,4}	{1,2,3,4}
C = A	{4}			
C = B	{4}			

A=4

• Dominio vacío.

Problema 4.-

1. Indicar si STRIPS resuelve el problema o no: Sí resuelve el problema ya que conseguimos vaciar la pila de objetivos.
2. En caso de que resuelva dar todos los operadores obtenidos en el plan en el orden en que se aplican.
    a. Secuencia de operadores: desapilar(F, G), dejar(F), coger(G), apilar(G, I), desapilar(G, I), dejar(G), desapilar(I, J), dejar(I), coger(J), apilar(J, G), desapilar(J, G), dejar(J), coger(G), apilar(G, I), coger(J), apilar(J, G)
3. ¿Es óptimo el plan?: No, ya que STRIPS dependiendo de las decisiones que se tomen en la busqueda, como el orden de descomposición de literales, puede no obtener el plan óptimo.

    Aquí se puede ver claramente como una de estas decisiones provoca no obtener el óptimo ya que al resolver primero 'SOBRE(G, I)' y después 'SOBRE(J, G)' al estar J debajo de I tendremos que deshacer los pasos realizados para cumplir la primera condición y este problema se repetirá dos veces en la traza de ejecución.

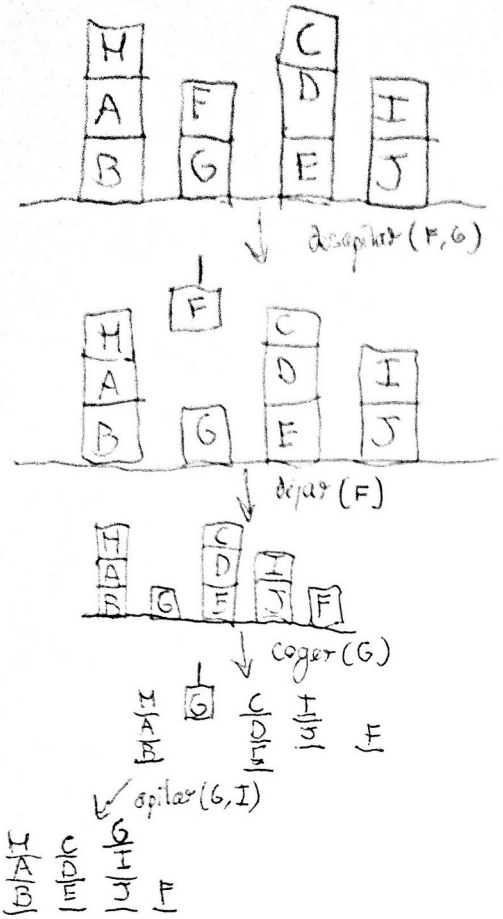
4. Aparte de los datos básicos anteriores, una descripción de cómo se ha resuelto el problema, y documentos gráficos de la resolución a mano.

    Para resolver el problema se ha simulado el planificador STRIPS con los operadores descritos, el estado inicial y una pila de objetivos que al inicio solo contiene el objetivo indicado.

    En un bucle infinito hasta que la pila de objetivos queda vacía se prueban en orden las acciones 1. Emparejar (comprobar si un predicado es cierto con el estado actual), 2. Descomponer (descomponer un predicado con múltiples literales de forma que se añade a la pila de objetivos cada uno de ellos de forma separada), 3. Resolver (se añade a la pila un operador que permite cumplir la condición si se aplica, para lo que tendrán que cumplirse sus precondiciones previamente), 4. Aplicar (se aplican los efectos del operador sobre el estado actual, eliminando los literales de la lista de supresión y añadiendo aquellos literales en la lista de adición).

    El plan que servirá para resolver el problema estará compuesto de los operadores en el orden en el cúal se han aplicado.

Descripción de Estado



Pila de objetivos

(Descripción) SOBRE(J,G) ∧ SOBRE(G,I)

~~SOBRE(J,G)~~

(Regla) ~~SOBRE(G,I)~~

~~apilar(G,I)~~

(Ins, ~~COGIDO(G) ∧ LIBRE(I)~~

(~~COGIDO(G)~~

(Emp) ~~LIBRE(I)~~

coger(G)

~~SOBREMESA(G) ∧ LIBRE(G) ∧ MANOVACIA~~

~~LIBRE(G)~~

~~SOBREMESA(G)~~

~~MANOVACIA~~

~~desapilar(X4,G)~~

Cierto con $X_1 = F$ ← ~~SOBRE(X4,G) ∧ LIBRE(X4) ∧ MANOVACIA~~

~~SOBREMESA(G)~~

~~LIBRE(G)~~

~~MANOVACIA~~

~~dejar(X2)~~

$X_2 = F$ — ~~cogido(X2)~~

apilar(J,G)

COGIDO(J) ∧ LIBRE(G)

~~COGIDO(J)~~

~~LIBRE(G)~~

coger(J)

SOBREMESA(J) ∧ LIBRE(J) ∧ Manovacía

MANOVACIA

~~LIBRE(J)~~

~~SOBREMESA(J)~~

desapilar(X_3, J)

$X_3 = I$ — SOBRE(X_3, J) ∧ LIBRE(X_3) ∧ Manovacía

Plan

desapilar(F,G)
dejar(F)
coger(G)
apilar(G,I)
desapilar(G,I)
dejar(G)
desapilar(I,J)
dejar(I)
coger(J)
apilar(J,G)
desapilar(J,G)
dejar(J)
coger(G)
apilar(G,I)
coger(J)
apilar(J,G)

1. Emparejar
2. Descomponer
3. Resolver
4. Aplicar

Pila de Objetivos

~~SOBRE(J,G) ∧ SOBRE(G,I)~~
~~apilar(J,G)~~
~~COGIDO(J) ∧ LIBRE(G)~~
~~coger(J)~~
~~SOBREMESA(J) ∧ LIBRE(J) ∧ MANOVACIA~~
~~MANOVACIA~~
~~desapilar(X_3,J)~~
~~SOBRE(X_3,J) ∧ LIBRE(X_3) ∧ MANOVACIA~~
~~MANOVACIA~~
~~SOBRE(X_3,J)~~
~~LIBRE(X_3)~~ , X_3=I
~~desapilar(X_4,I)~~
~~MANOVACIA ∧ LIBRE(X_4) ∧ SOBRE(X_4,I)~~ - cierto con X_4=G
~~dejar(X_5)~~
~~cogido(X_5)~~ - cierto con X_5=G
~~dejar(X_6)~~ -
~~cogido(X_6)~~ - cierto con X_6=I
~~SOBRE(J,G)~~
~~SOBRE(G,I)~~
~~apilar(G,I)~~
~~cogido(G)~~
~~libre(I)~~
~~coger(G)~~
~~SOBREMESA(G) ∧ LIBRE(G) ∧ MANOVACIA~~
~~MANOVACIA~~
~~LIBRE(G)~~
~~SOBREMESA(G)~~
~~desapilar(X_7,G)~~
~~MANOVACIA, LIBRE(X_7), SOBRE(X_7,G)~~ - Cierto con X_7=J
~~dejar(X_8)~~
~~cogido(X_8)~~ - X_8=J
~~apilar(J,G)~~
~~COGIDO(J) ∧ LIBRE(G)~~
~~COGIDO(J)~~
~~LIBRE(G)~~
~~coger(J)~~
~~SOBREMESA(J) ∧ LIBRE(J) ∧ MANOVACIA~~

Descripción de Estado (se muestran solo los nodos que cambian de estado en esta traza)

Se mantienen sin cambios

H	C	B
A	D	I
B	E	J

Se mantiene sin cambios F

↓ desapilar(G,I)

H	G
I	
J	

↓ dejar(G)

I	G
J	

↓ desapilar(I,J)

I	J	G

↓ dejar(I)

I	J	G

↓ coger(J)

I	G	J

↓ apilar(J,G)

J
G
I

↓ desapilar(J,G)

I	G	J

↓ dejar(J)

I	G	J

↓ coger(G)

I	G	J

↓ apilar(G,I)

G
I
J

↓ coger(J)

G	J
I	

↓ apilar(J,G)

Estado Final

H	C	J
A	D	G
B	E	I	F

Problema 5.-

1. Núcleo primero: MANOVACIA ^ LIBRE(E) ^ SOBRE(E, D) ^ SOBRE(D, C) ^ SOBRE(C, B)
2. Núcleo cuarto: COGIDO(D) ^ LIBRE(E) ^ LIBRE(C) ^ SOBRE(C, B)
3. Núcleo séptimo: SOBRE(D, E) ^ SOBRE(C, D)
4. ¿Qué literales aparecen en la casilla (0,3)? SOBRE(D, C)
5. ¿Qué literales aparecen en la casilla (0,4)? No aparece ninguno.
6. Aparte de los datos básicos anteriores, una descripción de cómo se ha resuelto el problema, y documentos gráficos de la tabla triangular obtenida.

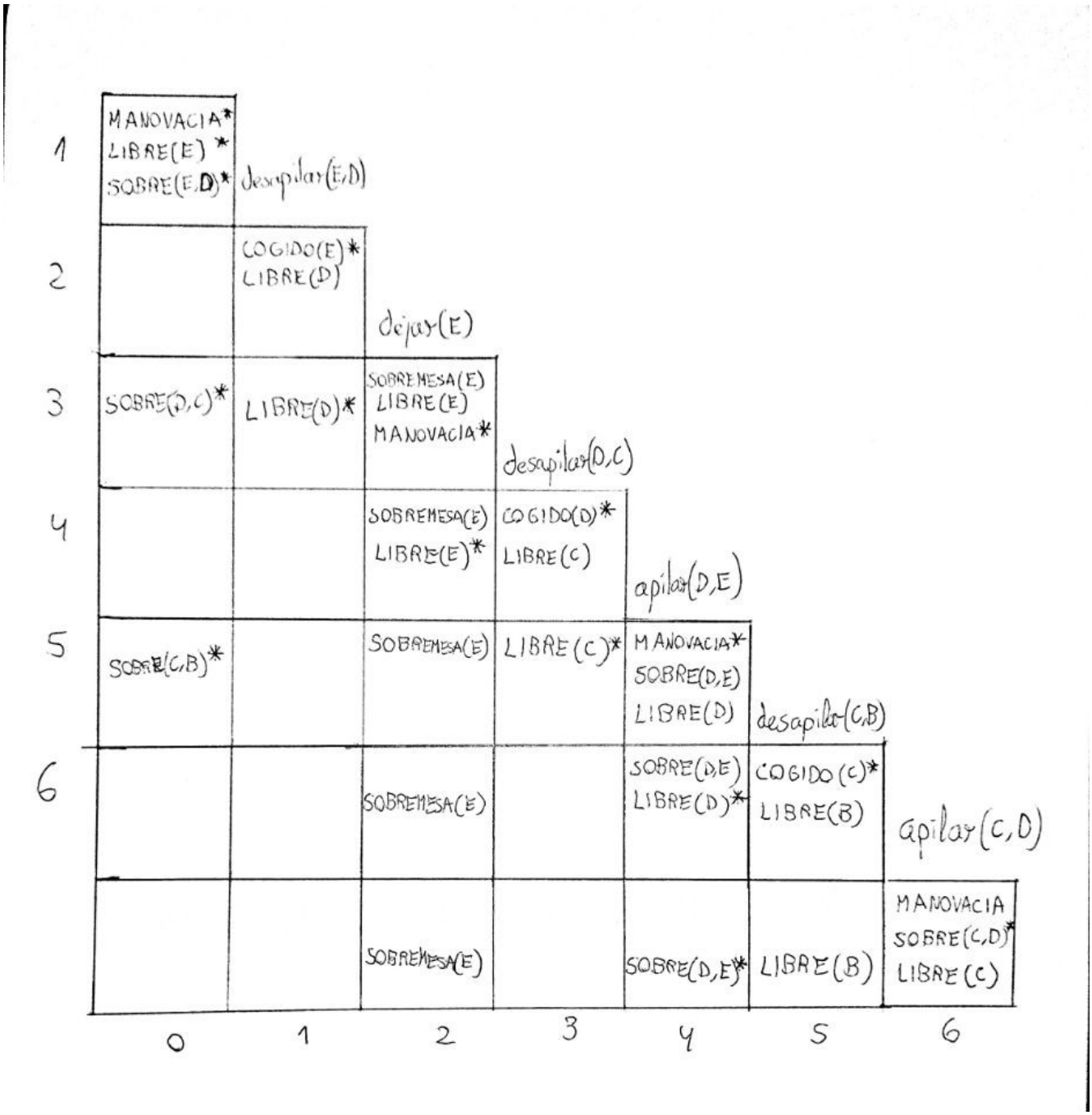
La tabla triangular que represente el plan estará compuesta de n+1 columnas y n+1 filas, siendo n el número de operadores del plan.

Tendremos que rellenar cada fila, las celdas a la izquierda de cada operador conteniendo literales que son las precondiciones de este y las celdas de debajo de cada operador conteniendo los literales (posiblemente repetidos) que son añadidos por el operador.

En la columna 1 tendremos que añadir los literales que son precondición para el operador i pero que no son añadidas por otro operador, por lo tanto son precondiciones del estado inicial.

En la última fila tendremos los literales que satisfacen la condición objetivo.

Rellenamos la tabla para conseguir estos resultados teniendo en cuenta los literales que añade cada operador y que literales necesita de precondición / elimina tras aplicarse.


1
2
3
4
5
6
MANOVACIA*
LIBRE(E)*
SOBRE(E,D)*
desapilar(E,D)
COGIDO(E)*
LIBRE(D)
dejar(E)
SOBRE(D,C)*
LIBRE(D)*
SOBREMESA(E)
LIBRE(E)
MANOVACIA*
desapilar(D,C)
SOBREMESA(E)
LIBRE(E)*
COGIDO(D)*
LIBRE(C)
apilar(D,E)
SOBRE(C,B)*
SOBREMESA(E)
LIBRE(C)*
MANOVACIA*
SOBRE(D,E)
LIBRE(D)
desapilar(C,B)
SOBREMESA(E)
SOBRE(D,E)
LIBRE(D)*
COGIDO(C)*
LIBRE(B)
apilar(C,D)
SOBREMESA(E)
SOBRE(D,E)*
LIBRE(B)
MANOVACIA
SOBRE(C,D)*
LIBRE(C)
0
1
2
3
4
5
6