UNIVERSIDAD CARLOS III DE MADRID

Asignatura: **INTELIGENCIA ARTIFICIAL**.
Campus: **LEGANÉS**

EVALUACIÓN CONTINUA 2014/15

NOTAS A TENER EN CUENTA ANTES DE LA REALIZACIÓN DEL EXAMEN

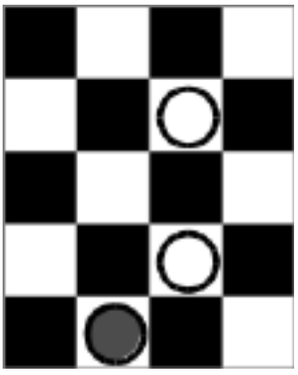
Antes de comenzar a responder a las preguntas lea detenidamente estas indicaciones:

1. El examen tendrá una duración de 1h45m.
2. No se permiten libros ni apuntes
3. Lea atentamente las preguntas fijándose con mucho detalle en la cuestión o cuestiones que se le plantean.
4. Ponga mucha atención al escribir sus datos personales.
5. No se responden dudas durante la celebración del examen
6. Habrá de esperar 30m antes de abandonar el aula una vez comenzado el examen.

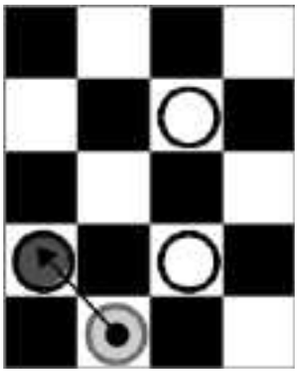
Ejercicio 1 (3 puntos)

Queremos representar con un sistema de producción un juego tipo damas simplificado. Hay dos jugadores, blanco y negro, sobre un tablero de 8x8 casillas, donde sólo se usan las casillas blancas. Las fichas de cada jugador empiezan a un lado del tablero, y sólo mueven hacia el lado contrario (hacia "delante"). Los movimientos son en diagonal únicamente.

En su turno cada jugador tiene dos opciones: 1) mover una ficha a una casilla que esté **delante a la izquierda o delante a la derecha** (fig.b) ; o 2) capturar una ficha enemiga saltando, **en esas direcciones,** por encima de la misma (fig c.), pero sólo si la casilla a la que salta está vacía. Si captura, puede seguir capturando con la misma ficha (tanto a derecha como a izquierda), hasta que no pueda continuar.



a) Ejemplo Posición



b) Movimiento

c) Captura

Usando cualquier notación que sea consistente y suficientemente detallada:

1. *Representa un estado cualquiera del juego mediante hechos. Si hay que crear muchos hechos, puedes indicar algunos ejemplos.*
2. *Añade los hechos necesarios para contemplar la alternancia entre jugadores una vez hayan concluido un turno.*
3. *Escribe detalladamente las reglas del juego, de forma que el juego pueda funcionar. Explica también en texto qué hace cada regla. No olvides indicar con qué mecanismo de resolución de conflictos tendría que ejecutarse tu sistema.*

Solución: Nota, usaremos resolución de conflictos en profundidad.

1. El tablero tiene una serie de casillas conectadas, pero las conexiones son diferentes para cada bando. Vamos a tener que distinguir direcciones en los movimientos.

   Identificamos las casillas con la notación (x,y) a partir de la esquina inferior izquierda, y podemos incluso añadir el estado en un solo hecho:

   Casilla(c21,negro), Casilla (c32,blanco), Casilla (c43,blanco), Casilla (c12,vacia), …

   Tendremos que saber qué casillas están “conectadas” en cada sentido. Podemos crear estos hechos a mano o mediante reglas:

   Conectada (arriba,izquierda,c12,c21)
   Conectada (abajo,derecha,c41,c32)
   …

2. Definiríamos los jugadores, sabiendo quién juega “hacia arriba” o “hacia abajo”:

   Jugador(blanco,arriba),Jugador(negro,abajo),
   Turno(negro),
   Después(negro,blanco),Después (blanco,negro)

3. Reglas que corresponden a los movimientos. Estrategia en profundidad, para que caso de empezar capturando, siga haciéndolo hasta no poder más.

R1 (capturar): Salta una ficha enemiga, moviendo a la casilla que hay detrás en el mismo sentido (LADO) y eliminando la enemiga que hay en medio (MEDIO). La casilla donde se acaba se guarda un hecho “Capturando(casilla)” que permite seguir capturando desde la posición donde se acaba (DEST).

R1.1: Para el primer movimiento de captura:

Turno(J) ∧ Jugador(J,DIR) ∧ Después (J,SIG) ∧ Casilla (POS,J) ∧
Conectada(DIR,LADO,POS,MEDIO) ∧ Casilla (MEDIO, SIG) ∧
Conectada(DIR,LADO,MEDIO,DEST) ∧ Casilla (DEST,vacia) ∧
~Capturando(DEST)
→
~ Casilla (POS,J), ~Casilla (MEDIO,SIG), Casilla (DEST,J),
Capturando(DEST)

R1.2: Para sucesivos movimientos de captura sólo se puede seguir desde el punto donde se estaba (Capturando(Casilla)):

Turno(J) ∧ Jugador(J,DIR) ∧ Después (J,SIG) ∧
Capturando (POS) ∧
Conectada(DIR,LADO,POS,MEDIO) ∧ Casilla (MEDIO, SIG) ∧
Conectada(DIR,LADO,MEDIO,DEST) ∧ Casilla (DEST,vacia)
→
~ Casilla (POS,J), ~Casilla (MEDIO,SIG), Casilla (DEST,J),
~ Capturando(POS), Capturando(DEST)

R2 (mover): Este movimiento se puede hacer sólo al principio del turno propio, donde asumimos que será falso el hecho Captura. Como sólo se puede hacer un movimiento, esta regla ya cambia el turno del jugador.

Juega(J) ∧ Jugador(J,DIR) ∧ Después (J,SIG) ∧
~Capturando(X) ∧ Casilla (POS,J) ∧
Conectada(DIR,LADO,POS,DEST) ∧Casilla (DEST,vacia)
→
~Casilla (POS,J), Casilla (DEST,J),
~ Juega(J) , Juega(SIG)

R3 (cambio de turno, prioridad más baja): Esta regla cambia el turno después de realizar todas las capturas posibles.

Juega(J) ∧ Jugador(J,DIR) ∧ Después (J,SIG) ∧ Capturando (POS),
→
~ Capturando (POS) , ~ Juega(J) , Juega(SIG)

Ejercicio 2 (2 puntos)

En un sistema de producción tenemos las siguientes **reglas**:

R_1(X): G ∧ S(x) → Z(x) ∧ retract (G)
R_2(X,Y): T(x) → G
R_3(X,Y): Q(x) ∧ R(y) ∧ P(y) → R(x) ∧ T(x) ∧ retract (Q(x))
R_4(X): P(x) ∧ R(x) → S(x)

Y se parte de los siguientes **hechos iniciales:** P(a), P(b), R(a), Q(a), Q(b)

1. *Supón que el sistema opera con una estrategia de resolución de conflictos con la estrategia "**primero la más nueva**" **(profundidad).** Muestra la secuencia de reglas ejecutadas y datos en la base de hechos en una tabla con el formato de abajo. En AGENDA escribe las instancias completas, en EJECUTA simplemente marca la instancia que se ejecuta. **(Nota**: Si queda duda en algún punto**,** la regla 1 tiene prioridad sobre la regla 2, luego va la 3, y luego la 4).*
2. *Indique cuál es el contenido de la base de hechos cuando el sistema se detiene.*
3. *Si se ejecutara con la estrategia **"primero la más antigua" (amplitud)**, ¿habría alguna diferencia en el estado final?*

PASO	HECHOS NUEVOS (+assert, -retract)	AGENDA (escribe la instancia completa)	EJE-CUTA
1	P(a), P(b), R(a), Q(a), Q(b)	Q(a) ∧ R(a) ∧ P(a) → R(a) ∧ T(a) ∧ retract (Q(a)) Q(b) ∧ R(a) ∧ P(a) → R(b) ∧ T(b) ∧ retract (Q(b)) P(a) ∧ R(a) → S(a)	X
2	-Q(a),+T(a)	T(a) → G Q(b) ∧ R(a) ∧ P(a) → R(b) ∧ T(b) ∧ retract (Q(b)) P(a) ∧ R(a) → S(a)	X
3	+G	Q(b) ∧ R(a) ∧ P(a) → R(b) ∧ T(b) ∧ retract (Q(b)) P(a) ∧ R(a) → S(a)	X
4	-Q(b),+R(b) +T(b)	T(b) → G P(b) ∧ R(b) → S(b) P(a) ∧ R(a) → S(a)	X
5	(nada, G ya está en la BH)	P(b) ∧ R(b) → S(b) P(a) ∧ R(a) → S(a)	X
6	+S(b)	G ∧ S(b) → Z(b) ∧ retract (G) P(a) ∧ R(a) → S(a)	X
7	-G,Z(b)	P(a) ∧ R(a) → S(a)	X
	+S(a)		FIN

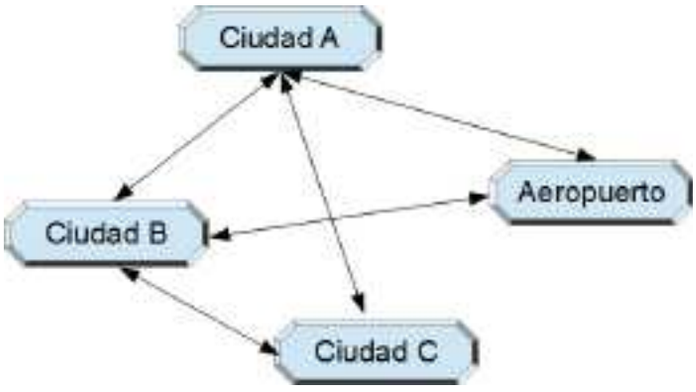
Resultado: P(a),P(b),R(a),T(a),R(b),T(b),S(b),S(a),Z(b)

En caso de más antigua aparecerá un S(a) que con G dará Z(a) y luego aparecerá G, S(b) y Z(b).

Ejercicio 3 (2 puntos)

Se pretende resolver el problema siguiente mediante búsqueda:

Un vehículo robótico parte del aeropuerto, debe recoger a varias personas en sus ciudades y llevarlos al aeropuerto de nuevo. Hay **M** ciudades parcialmente interconectadas por carreteras. El vehículo tiene capacidad para **N** pasajeros y sólo puede descargarlos en el aeropuerto. El objetivo es llevar a todos los pasajeros al aeropuerto haciendo el menor número de trayectos posible. En la figura puedes ver un ejemplo de instancia de este problema.



Pasajero	Situación
Pepe	Ciudad A
Rosa	Ciudad B
Eva	Ciudad C

Ejemplo de instancia del problema

1. ***Define** cómo representarías los estados del problema general (ojo, no esta instancia en particular). Considera que puede haber más ciudades que personas, y varias personas en una misma ciudad.*
2. *En particular **escribe** cuál sería el estado inicial para el problema que se muestra en la figura. ¿Cuál sería el estado final?*
3. ***Especifica** con claridad los operadores del problema, incluyendo sus posibles restricciones, en función del número **N**.*
4. ***Define** una heurística admisible para este problema, justificando por qué lo sería*
5. *Sobre el ejemplo de la figura, **genera** el árbol de búsqueda resultado de expandir el nodo inicial (únicamente)*

Representación (1):

Tendremos que saber dónde está el vehículo, y dónde cada pasajero. Podemos por ejemplo decir que un estado lo denotamos como una lista cuya longitud es número de pasajeros+1 **(Ciudad-Vehiculo, Ciudad-P1, Ciudad-P2, Ciudad-P3...)** donde cada elemento de la lista es una ciudad. Usamos “V” para indicar que un pasajero está en el vehículo. Sólo puede haber N letras V.

Estados que piden:
Inicio: (Aeropuerto,A,B,C)
Fin: (Aeropuerto,Aeropuerto,Aeropuerto,Aeropuerto)

Operadores: Podríamos usar Mover, Cargar y Descargar, pero como sólo importan los movimientos, los dos últimos no van a aportar nada. De manera que podemos usar un solo operador: Mover (Destino):

Mover(Destino):
Requisito: el vehículo está en Origen y la ciudad Origen y Destino están conectadas
Efectos:

Destino distinto de Aeropuerto
(**Origen**, Ciudad-P1,Ciudad-P2, …) → (**Destino**, Ciudad-P1, Ciudad-P2, …)

y carga los pasajeros (hasta un máximo de N)

(**Destino**, **Destino**, Ciudad-P2, **Destino**, …) → (**Destino**, **V**, Ciudad-P2, **V**, …)
(Siendo el número de "V" menor o igual a N)

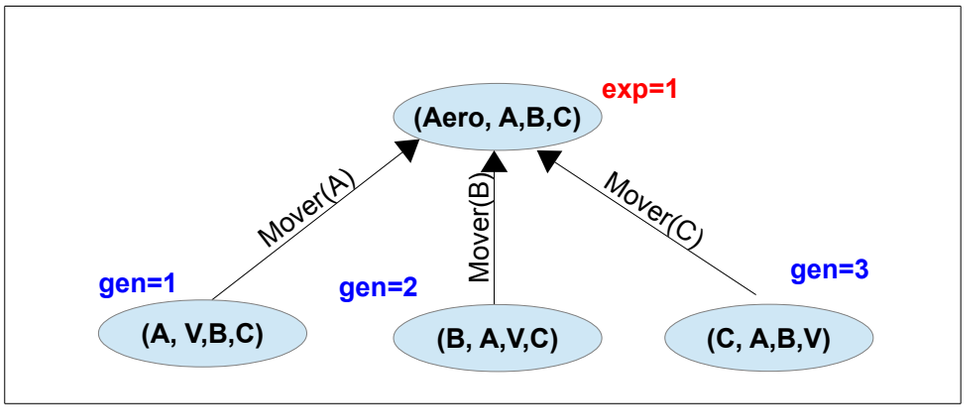
Destino es Aeropuerto

(**Origen**, Ciudad-P1,Ciudad-P2, …) → (Aeropuerto, Ciudad-P1, Ciudad-P2, …)

Y descarga:

(Aeropuerto, **V**, Ciudad-P2, **V**, …) → (Aeropuerto, Aeropuerto, Ciudad-P2, Aeropuerto, …)

Heurística:
Podemos contar 1 por cada ciudad distinta del aeropuerto que aparece en el estado, sin contar el token "V".



Representación 2:

Hay una representación cómoda si nos damos cuenta de que los pasajeros no se distinguen los unos de los otros.

(Ciudad-Vehículo, #Pjros-Cargados, #Pjros-Ciudad-A, #Pjros-Ciudad-B,...)

Inicio: (Aeropuerto,0,1,1,1)
Fin: (Aeropuerto,0,0,0,0)

Mover (D): Si **Origen** (ciudad del vehículo) y D están conectadas, y pD es el número de pasajeros en D,

Si D distinto de Aeropuerto

incr=min(N-p0-pD,pD) (incremento de viajeros en el vehículo)

(**Origen**,p0,p1,p2,p3,...,pD, …) → (D,**p0+incr**,p1,p2,p3,...,**pD-incr**, …)

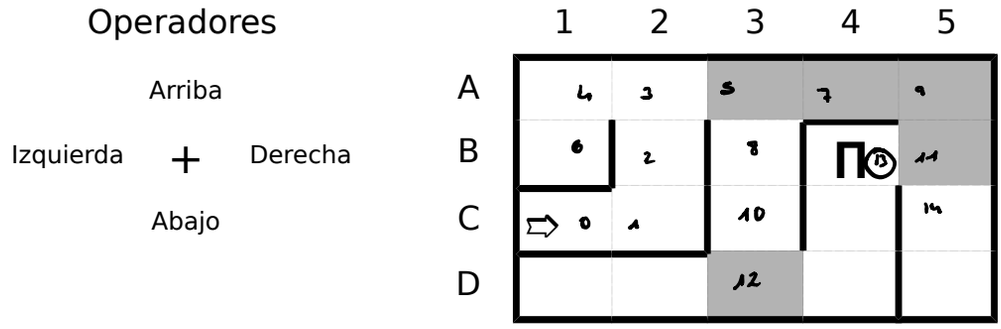
Si D es Aeropuerto

(**Origen**,p0,p1,p2,p3,...) → (Aeropuerto,**0**,p1,p2,p3,...)

Una heurística admisible podría ser el número de valores *pl* de la lista que no valen 0. Se podría añadir también el número total de pasajeros (suma de *pl*) divido entre N (redondeado hacia abajo) ya que si el número es mayor que N, hacen falta más viajes.

Ejercicio 4 (3 puntos)

Queremos recorrer el siguiente laberinto, donde los trazos gruesos simbolizan paredes no atravesables, las casillas blancas se tardan **2 minutos** en recorrer (es decir suman 2 min al tiempo total), y las casillas grises son tramos complicados que requieren **6 minutos**. Sólo se puede mover en dirección **vertical u horizontal**. La salida del laberinto está representada por el portal en B4, y la posición inicial es C1. El objetivo es llegar a la salida en el mínimo tiempo posible. Generamos nodos en orden: izquierda, arriba, derecha, abajo.

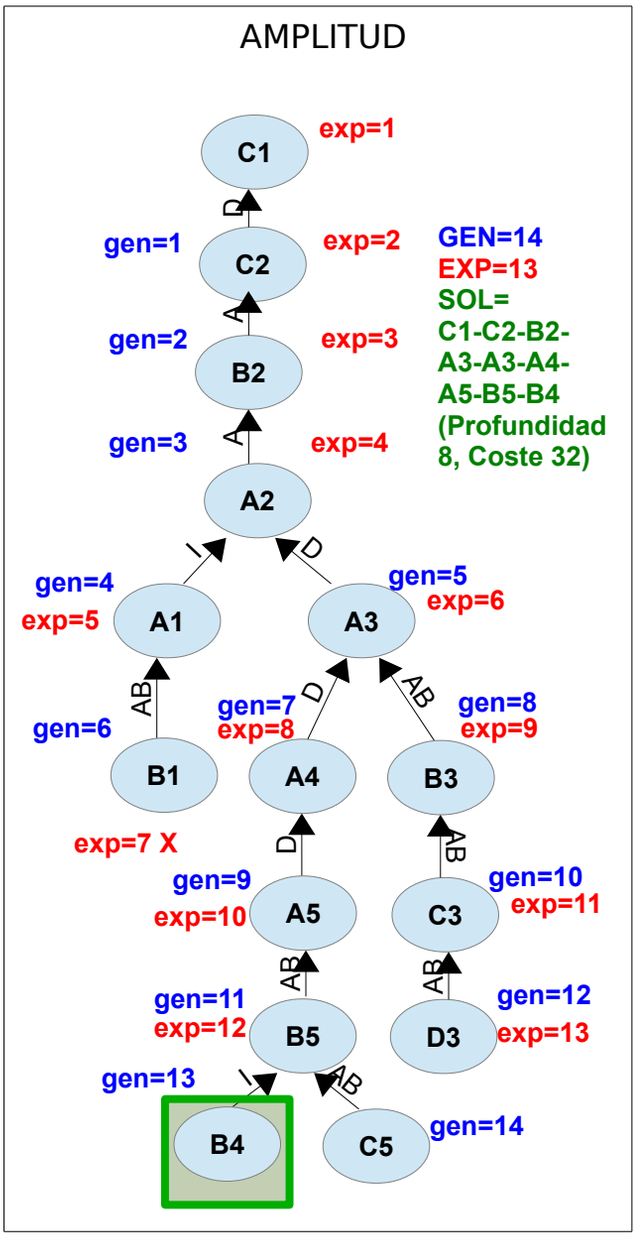


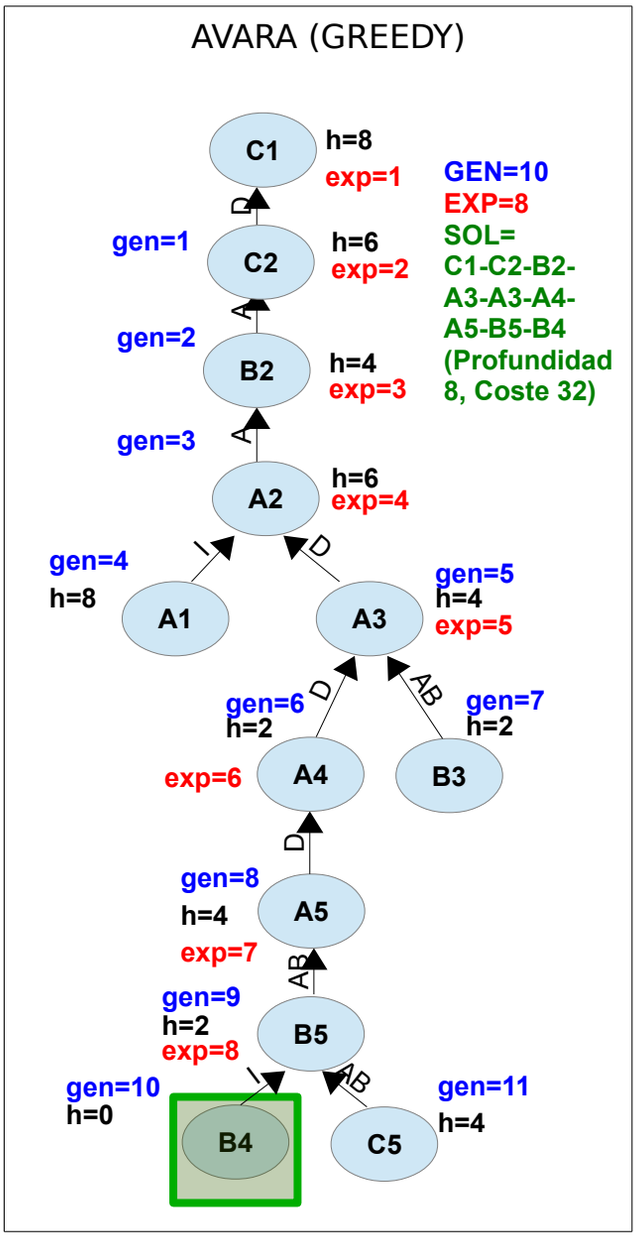
1. *Ejecuta búsqueda en amplitud ¿Cuál es la profundidad de la solución encontrada?*
2. *Define una heurística para este problema que le parezca adecuada, mejor cuanto más informada.*
3. *Con la heurística indicada, ejecuta búsqueda avara (greedy). Indica en el árbol el orden en que los nodos se generan. Indica la solución alcanzada, su coste, y el número de nodos generados y expandidos.*
4. *Con la heurística y costes indicados, ejecuta búsqueda en A*. Indica los mismos valores que en el apartado anterior.*
5. *¿Qué algoritmo tiene un coste computacional mayor, si se mide en número de nodos expandidos? Explica si crees que el grado de información de la heurística escogida en el punto 2 tiene un efecto en este valor para alguno de los algoritmos.*

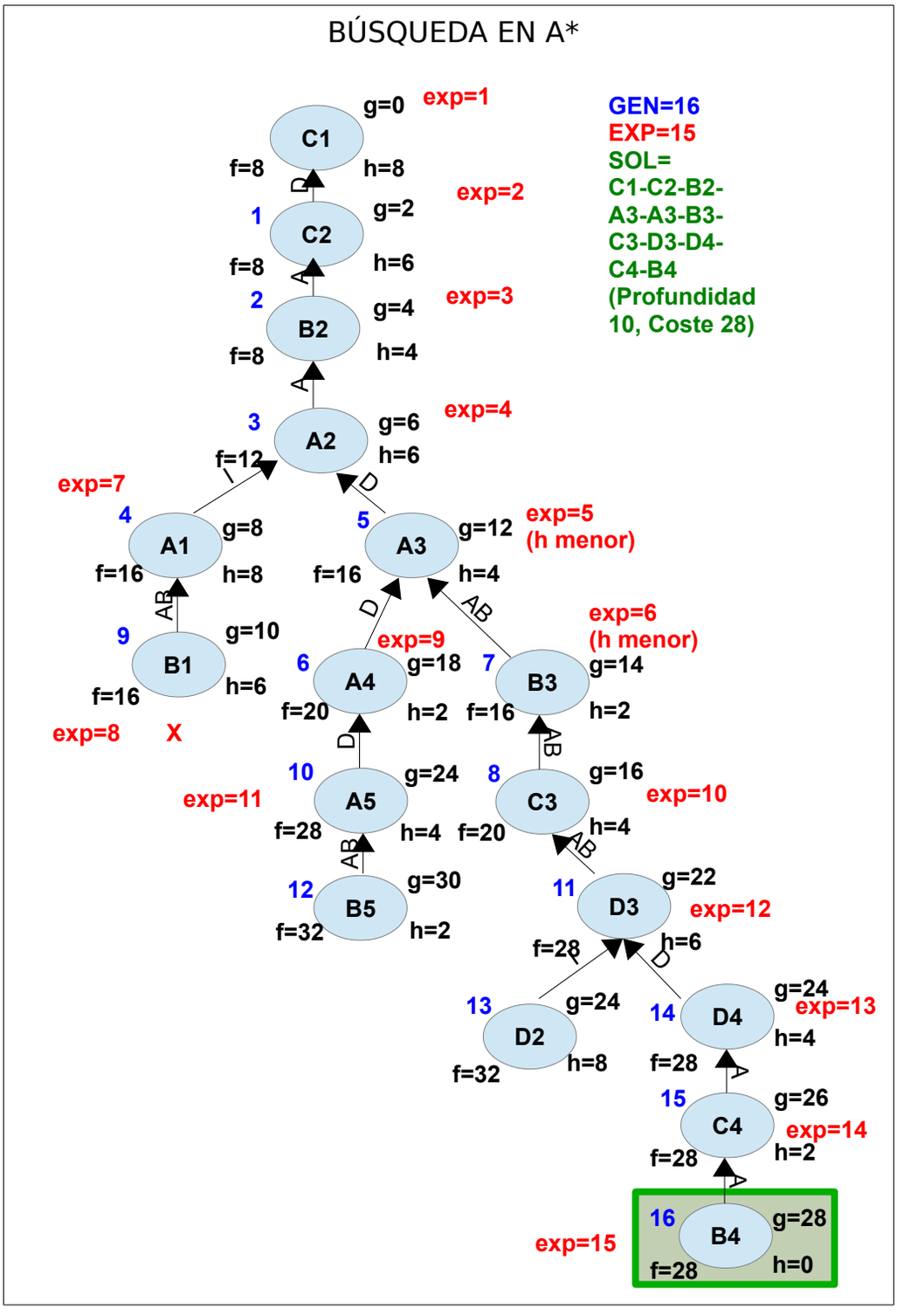
1. (ver figura) La profundidad de la solución en Amplitud es 8. El algoritmo genera 14 nodos, de los que expande 13.
2. La heurística podría ser la distancia de Manhattan a la salida, pero como el coste mínimo es 2, podemos multiplicarla por este factor:

$$h(n) = 2 \times \text{Manhattan}(n \rightarrow B4)$$

3. (ver figura) La búsqueda avara obtiene una solución de coste 32 generando 10 nodos y expandiendo 8.
4. (ver figura) La búsqueda en A* obtiene una solución de coste 28 generando 16 nodos y expandiendo 15.
5. La heurística utilizada permite a A* descartar algunos nodos (D2, B5). Con una heurística menos informada tal vez se hubieran tenido que expandir estos.


AMPLITUD
C1
exp=1
D
gen=1
C2
exp=2
A
gen=2
B2
exp=3
gen=3
A
exp=4
A2
I
D
gen=4
exp=5
A1
A3
gen=5
exp=6
AB
gen=6
B1
D
gen=7
exp=8
A4
AB
gen=8
exp=9
B3
exp=7 X
D
AB
gen=9
exp=10
A5
C3
gen=10
exp=11
AB
AB
gen=11
exp=12
B5
D3
gen=12
exp=13
gen=13
I
AB
B4
C5
gen=14
GEN=14
EXP=13
SOL=
C1-C2-B2-
A3-A3-A4-
A5-B5-B4
(Profundidad
8, Coste 32)


AVARA (GREEDY)
C1
h=8
exp=1
D
gen=1
C2
h=6
exp=2
A
gen=2
B2
h=4
exp=3
gen=3
A
h=6
exp=4
A2
I
D
gen=4
h=8
A1
A3
gen=5
h=4
exp=5
gen=6
h=2
D
AB
gen=7
h=2
exp=6
A4
B3
D
gen=8
h=4
exp=7
A5
AB
gen=9
h=2
exp=8
B5
I
AB
gen=10
h=0
B4
C5
gen=11
h=4
GEN=10
EXP=8
SOL=
C1-C2-B2-
A3-A3-A4-
A5-B5-B4
(Profundidad
8, Coste 32)


BÚSQUEDA EN A*
GEN=16
EXP=15
SOL=
C1-C2-B2-
A3-A3-B3-
C3-D3-D4-
C4-B4
(Profundidad
10, Coste 28)
C1
g=0
f=8
h=8
exp=1
D
C2
1
g=2
f=8
h=6
exp=2
A
B2
2
g=4
f=8
h=4
exp=3
A
A2
3
g=6
f=12
h=6
exp=4
I
D
A1
4
g=8
f=16
h=8
exp=7
A3
5
g=12
f=16
h=4
exp=5
(h menor)
AB
B1
9
g=10
f=16
h=6
exp=8
X
D
AB
A4
6
g=18
f=20
h=2
exp=9
B3
7
g=14
f=16
h=2
exp=6
(h menor)
D
AB
A5
10
g=24
f=28
h=4
exp=11
C3
8
g=16
f=20
h=4
exp=10
AB
AB
B5
12
g=30
f=32
h=2
D3
11
g=22
f=28
h=6
exp=12
D
D2
13
g=24
f=32
h=8
D4
14
g=24
f=28
h=4
exp=13
A
C4
15
g=26
f=28
h=2
exp=14
A
B4
16
g=28
f=28
h=0
exp=15