Departamento de Informática
Universidad Carlos III de Madrid

Grado en Ingeniería en Informática

Artificial Intelligence
Partial exam 1

March 2015

General indications

- Time assigned to the exam is **2 hours**
- You cannot leave the classroom during the exam, unless you have finished it
- Exams cannot be answered using a pencil

Exercise 1 (2.5p)

Consider the graph in Fig. 1, where S is the initial state and $G1$, $G2$, and $G3$ are goal states. Labels on arcs represent the cost of traversing them. Numbers on nodes represent the heuristic value to reach a goal state from the corresponding node.

Expand the search tree to obtain a path from S to a goal state for the following cases: (1) Hill Climbing search (0.5p); (2) Dijkstra search (0.5p); (3) A* search (0.5p). Indicate beside each node its generation and expansion orders and all the necessary information to interpret correctly your solution. Check for repeated states and prune them whenever possible. Follow an alphabetical order to generate successors.

Answer the following questions:

1. (0.5p) Which of the above algorithms finds the optimal solution? Which one guarantees the optimal solution? Explain your answers.
2. (0.5p) In your opinion, which algorithm is more adequate to solve this kind of problems? Explain your answer.

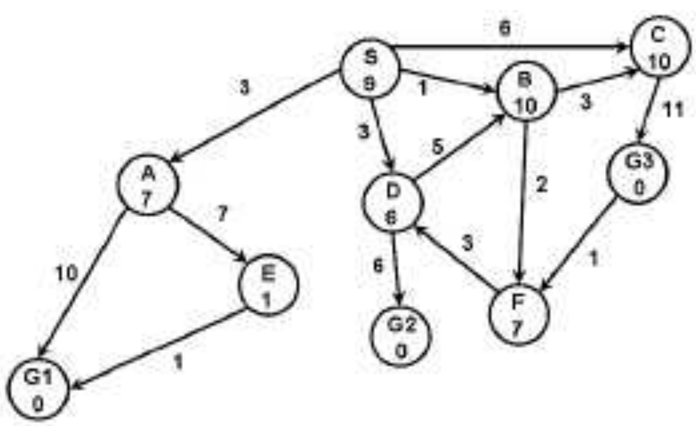


Fig. 1: Graph for exercise 1.

Exercise 2 (2.5p)

In a maze, as the one showed in Fig. 2, there is a car (denoted with an arrow) that has to be parked at a position X. The car is facing some direction $d \in \{N, S, E, W\}$. For instance, in the situation in Fig. 2 the direction is *N (north)*. Also, the car has a velocity. The velocity at a given moment denotes the number of cells that the car can move. The initial velocity is zero.

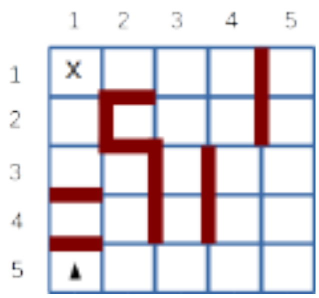


Fig. 2: Car problem.

The car can execute the following actions:

- Change the direction: the car changes the direction by 90 degrees either clockwise or counterclockwise. The direction can only be changed when the velocity is zero.
- Advance: the car advances as many cells as its current velocity in its current direction. Advance actions that lead to a collision with a wall are illegal. This action can be executed in 3 modalities: *increase*, *decrease* or *maintain*. These modalities determine the velocity in the reached state either by increasing by one, decreasing by one or maintaining the velocity in the source state. The velocity should remain in the interval $[0, V_{max}]$, where V_{max} is defined before solving the problem. When the velocity is zero, to advance maintaining the velocity is also illegal.

The goal is to park the car using as few actions as possible.

1. (0.5p) Represent the state space for this problem.
2. (0.5p) Represent the initial state and the goal state.
3. (0.5p) Represent the operators of this problem including their applicability conditions and result. What is the cost function?
4. (0.5p) What of the studied algorithms will guarantee the optimal solution?
5. (0.5p) Could you propose an admissible heuristic function for this problem?

Exercise 3 (2.5p)

Suppose we want to build a production system for the problem in Exercise 2.

1. (1.25p) Use predicate logic to represent the information of the Working Memory for that problem. What is the initial Working Memory?
2. (1.25p) Explain how to model the action advance and include an example.

Exercise 4 (2.5p)

Given the following rules of a production system:

R1 (priority 5): IF a(X,Y), a(X+1,Z) b(Y,Y1) b(Z, Z1)
THEN a(X,Y1), a(X+1, Z1), ¬ a(X,Y), ¬ a(X+1,Z)

R2 (priority 10): IF a(X,1), a(X+1, 1), a(X+2, 1), a(X+3,1)
THEN stop_exec()

where

- Capital letters represent variables.
- Instantiations with a higher priority are preferred.
- stop_exec() is a special directive to stop the execution.

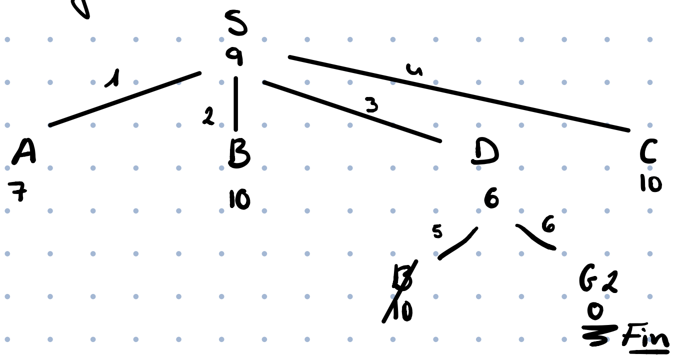
Simulate a possible execution of the production system for a random conflict resolution strategy. Assume the initial working memory is:

$$\text{WM}_0 = \{a(1,1), a(2,1), a(3,2), a(4,2), b(1,2), b(2,1)\}$$

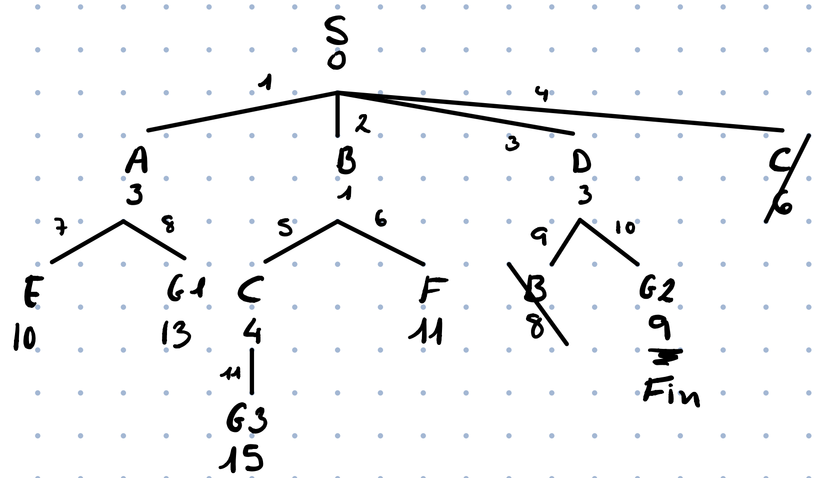
For each cycle, show clearly the conflict set, the selected rule and the resulting working memory.

1.) Expand:

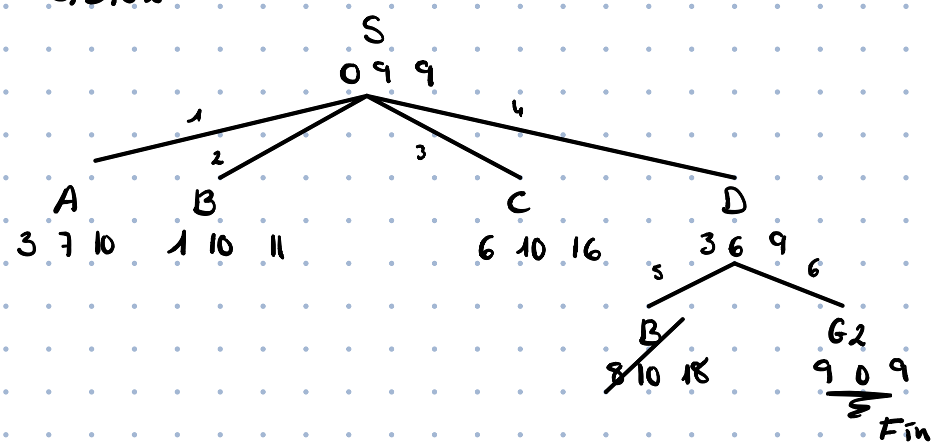
1. Hill - Climbing: S, D, G2



2. Dijkstra: S, D, G2



3. A*: S, D, G2



1. Todos encuentran la solución optima, aunque 1 con más ops, pero solo la aseguran A*, con heurística admisible, y Dijkstra, que siempre la encuentra. Apesar de que la h ha guiado bien a HC

2. A*, ya que la heurística que usa es admisible por lo que siempre encuentra la solución más optimas, y lo hace con menos operaciones que Dijkstra.

2.) $\overset{\uparrow}{N}$ $V_0 = 0 \frac{\text{cells}}{\text{move}}$ Cambiar dir si $V=0$.

$V \in (0, V_{max})$ Avanzar solo si puede, no chocar. < Aumentar $V += 1$ $V < V_{max}$
Mantener V $V \neq 0$
Reducir $V -= 1$ $V > 0$

1. car(x, y, V, h)
   x: pos x $\in (1,5)$
   y: pos y $\in (1,5)$
   V: velocidad $V \in (0, V_{max})$
   h: horientación $h \in (N, S, E, W)$

hori(N, E)
hori(E, S)
hori(S, W)
hori(W, N)

pared(A, B, C, D)
A: pos x $\in (1,5)$
B: pos y $\in (1,5)$
C: Horientacion $\in (N, S, E, W)$
D: Nº casillas puedo avanzar

2. inicial: car(1, 5, 0, N)

   final: car(1, 1, V, H) $V \in (0, V_{max}$
   $H \in (N, S, W, E)$

3. CambDir Hori: car(X, Y, 0, H) hori(H, F) $\longrightarrow$ modify car(X, Y, 0, F)

   Camb Dir Anti: car(X, Y, 0, H) hori(F, H) $\longrightarrow$ modify car(X, Y, 0, F)

   AvanAu N: car(X, Y, V, N) pared(X, Y, N, V) $V < V_{max}$ $\rightarrow$ modify car(X, Y+V, V+1, N)

   ⋮

   Avan Man N: car(X, Y, V, N) $V > 0$ pard(X, Y, N, V) $\longrightarrow$ modify car(X, Y+V, V, N)

   ⋮

   AvanRe N: car(X, Y, V, N) $V > 0$ pared(X, Y, N, V) $\xrightarrow{\text{mod}}$ car(X, Y−V, V−1, N)

   ⋮

4. Amplitud, Dijkstra y A^* con $h(n)$ admisible.

5. $h(n)$ = Distancia Manhattan = $\max(|x_g - x_i|, |y_i - y_g|)$

4.)

Busco $a(1,1)^{*}$
$a(2,1)^{*}$
$a(3,1)$
$a(4,1)$

$WM_0 = \{ a(1,1), a(2,1), a(3,2), a(4,2), b(1,2), b(2,1) \}$

$CC_0 = \{ R1(x=1, y=1, z=1, y1=2, z1=2),$

$R1(x=2, y=1, z=2, y1=2, z1=1)$

$R1(x=3, y=2, z=2, y1=1, z1=1) \} \rightarrow$ Ejecutar

$WM_1 = \{ a(1,1), a(2,1), b(1,2), b(2,1), a(3,1), a(4,1) \}$

$CC_1 = \{ R1(x=1, y=1, z=1, y1=2, z1=2),$

$R1(x=2, y=1, z=2, y1=2, z1=1),$

$R2(x=1) \} \rightarrow$ Ejecutar

$WM_2 = \{ a(1,1), a(2,1), a(3,1), a(4,1), b(1,2), b(2,1) \}$

STOP