

# Search problems

Samy Aittahar

ULiege - INFO0049

Tuesday 27<sup>th</sup> February, 2018

# Search problem

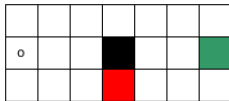
A search problem is defined by the following components :

- ▶ A set of state, which also includes :
  - ▶ An initial state,
  - ▶ A subset of forbidden states.
- ▶ A set of available actions per state,
- ▶ A transition model which maps a state-action pair to another state,
- ▶ A goal test.

# Definitions

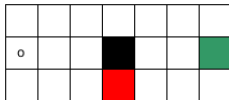
- ▶ A state space is the set of all states reachable from the initial state using the transition model,
- ▶ A path is a sequence of state generated by the transition model with an available action for each state,
- ▶ A solution is a path which satisfies the goal test, which generally accept paths ending with a (set of) particular state(s) (alias *final state(s)*).
  - ▶ A numeric value can be assigned to a solution to represent its cost,
  - ▶ If the solution achieves the lowest possible cost, then it is *optimal*.
  - ▶ Note that a path which contains a cycle is generally not a solution. A possible solution is to forbid to revisit any state integrated in the path.

## Example



- ▶ State space :  $S = \{(i,j) \in \mathbb{N}_1^2 \mid \forall i \leq 3, j \leq 7\} \setminus (2,4),$
- ▶ Initial state :  $(2,1),$
- ▶ Available actions :
  - ▶  $U = \{(0,1), (1,0), (-1,0), (0,-1)\}$  for all states except for  $(2,3), (2,5), (1,4)$  and  $(3,4),$
  - ▶  $U \setminus (0,1)$  for state  $(2,3), U \setminus (0,-1)$  for state  $(2,5).$
- ▶ Final state :  $(2,7),$
- ▶ Forbidden state :  $(3,4).$

## Example (cont'd)



- ▶ Transition model :

$$t((i,j), (x,y)) = \begin{cases} (i+x, j+y) & \text{if } (i+x, j+y) \in S \setminus (2,4) \\ (i,j) & \text{otherwise} \end{cases}$$

- ▶ Transition cost is 1 everywhere.

# Paths

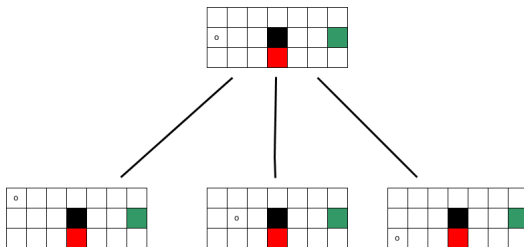
(not exhaustive)



- ▶ Optimal path solution
- ▶ Path solution
- ▶ Not a path solution

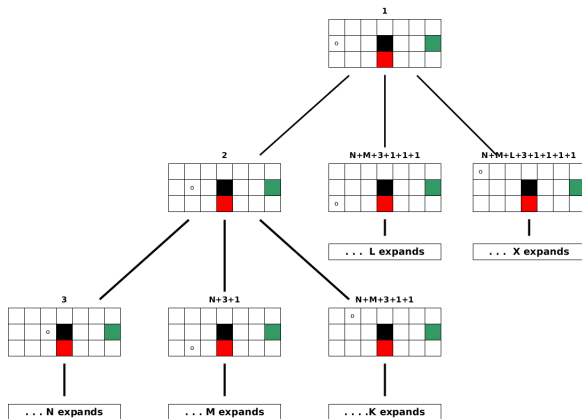
## Searching over a tree

⚠ do not confuse with search tree in prolog !



- ▶ *Node expansion* leads to a successor state per action.
- ▶ Return the first path found which satisfies the goal test
- ▶ Which node should be expanded first ?

## Naive approach (1) - Depth First Search



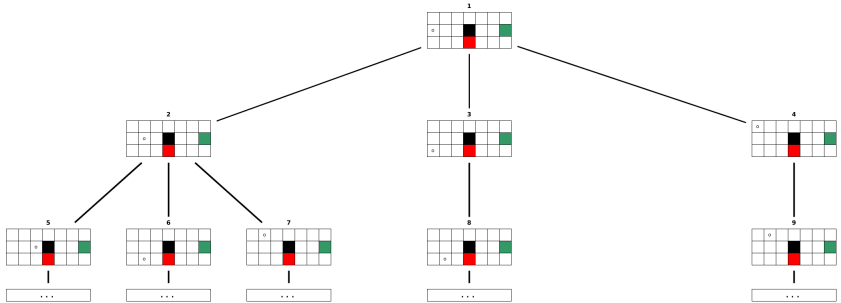
*Node labels define expansion order*



## Naive approach (1) - DFS (cont'd)

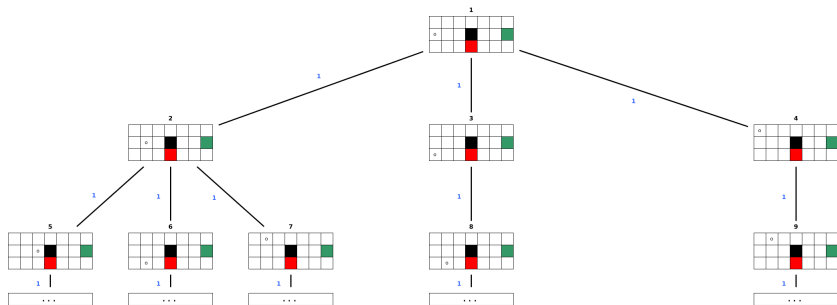
- ▶ Node depth defines expansion order.
- ▶ Properties ? Issues ?

## Naive approach (2) - Breadth First Search



- ▶ Node depth defines expansion order.
- ▶ Properties ? Issues ?

## Naive approach (3) - Uniform Cost Search (UCS)



- ▶ Node path cost defines expansion order.
- ▶ Properties ? Issues ?

## What is missing ?

- ▶ Previous methods are *uninformed*.
- ▶ How to provide a generic guidance for the expansion order ?

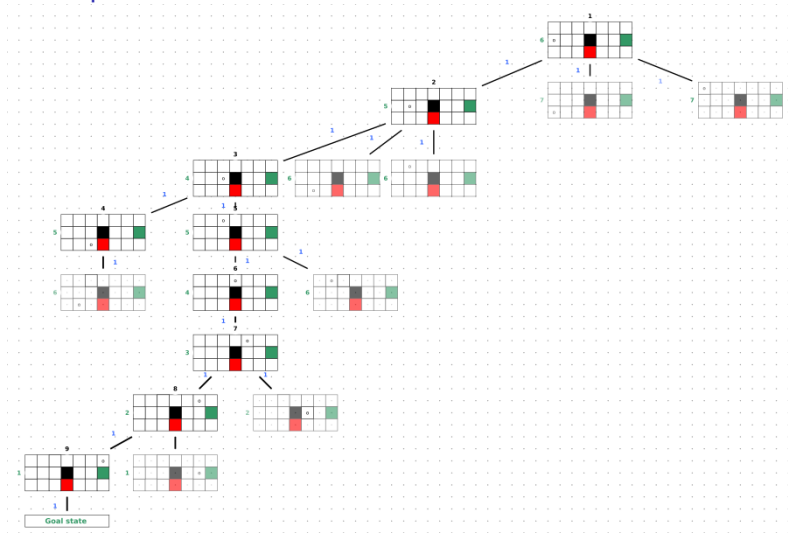
## State evaluation

- ▶ Say we define a heuristic which is a state evaluation function  $h: S \rightarrow \mathbb{R}_+$
- ▶ Ideally, we would like a function  $h^*(x)$  which gives the lowest achievable cost by reaching the goal state from  $x$ .
- ▶ Unfortunately, we are unlikely to have this function. Are we done ? **No !**

# A\*

- ▶  $A^* = \text{UCS} + \text{heuristic}$
- ▶ We can have an approximation of  $h^*$ , as long as it is admissible
  - ▶ Let  $h^0$  be the 0-everywhere heuristic
  - ▶ An heuristic  $h$  is admissible if  $h^0 \leq h \leq h^*$
  - ▶ Note that  $h^0$  reduces  $A^*$  to UCS
- ▶ Manhattan distance is a well known heuristic
  - ▶  $d(A, B) = |X_b - X_a| + |Y_a - Y_b|$
- ▶ Quality of the heuristic will directly influence the number of expanded nodes.

# A\* example

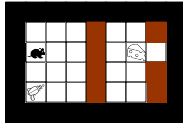


## Remarks

- ▶ This kind of search problem can also be addressed through a Traveling Salesman Problem solver-like.
  - ▶ In one sentence, looks for the shortest path that crosses each node (at most) once.
  - ▶ NP-Complete problem. Many greedy algorithms and integer programming can be used to approximate the optimal solution.
- ▶ What is something that tries to prevent us to reach the objective ? This is *adversarial search*. We'll see it later.



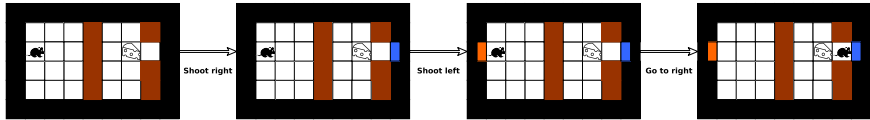
## Homework : Aperture maze



*Goal : collect the cheese(s) without crossing a red cell*


- ▶ The mouse can move left, right, up or down.
- ▶ When equipped with a gun, the mouse can also shoot a bullet left, right, up or down.
- ▶ Once a bullet reaches a wall, it creates a portal oriented in the opposite direction.
- ▶ At most two portals can be created. If a third portal is created, the first one directly vanish.
- ▶ If the mouse reaches a portal, it lands directly to the adjacent non-wall cell.

## Homework : Motion example



*For simplicity here we assume that the mouse has already grabbed the gun.*

## Homework : Input/Output

- ▶ The input is a list of list of the following symbols :
  - ▶ '%' is a wall,
  - ▶ 'r' is a red cell,
  - ▶ 'm' is the mouse,
  - ▶ 'g' is the gun,
  - ▶ 'c' is the cheese,
  - ▶ ' ' is a white cell
- ▶ The output is a list of the actions below
  - ▶ 'l','r','u','d' respectively for left,right,up,down,
  - ▶ 'L','R','U','D' respectively for shooting at left, right, up, or down
- ▶  Ensure that your path meets the goal test.

## Homework : Deliverable and deadline

Expected : a single Prolog file, to be sent via the Montefiore Submission Platform in the indicated deadline. This file needs to contains the following parts :

- ▶ A formalization of the Aperture Maze search problem.
- ▶ The source code of your implementation of an algorithm presented here
  - ▶ It needs to contains a main predicate *solve*(*M*,*O*) where *M* is the input and *O* is the output as specified in the previous slide.
  - ▶ You may consider other algorithms but be careful to explain them clearly and why you used them.
- ▶ Some executions examples with solution/computation time
  - ▶ Use the predicate *time*(*P*) with *P* your predicate call (it needs to be named *solve*).

## Homework : Evaluation

- ▶ Formalization of the search problem.
- ▶ Structure and clarity of your code.
  - ▶ A clear and concise explanation of your solving strategy is expected.
  - ▶ There is no expected report so be really careful about the documentation.
- ▶ Performance (computation time and quality of the solution).
  - ▶ Your solution will be tested on big grids.
- ▶ Oral presentation.
  - ▶ Scheduled for 27/03/2018.

That's all folks

If you want to know more about AI :

*Stuart J. Russell and Peter Norvig. 2003. Artificial Intelligence: A Modern Approach (2 ed.). Pearson Education.*