FACULTY OF
COMPUTER SCIENCE

IA@FII

Artificial Intelligence

3rd year, 1st semester

State based models for decision problems

Week 3: informed strategies and alternative problem spaces

"In any moment of decision, the best thing you can do is the right thing, the next best thing is the wrong thing, and the worst thing you can do is nothing." Theodore Roosevelt

Let’s find a NP-complete problem and a computer able to solve it

Problem: finding a path in a graph

Describing a model - reducing a problem

- describe a state
- identify special states and the problem space
- describe the transitions and validate them
- specify a search strategy

Search strategies

- Uninformed - no distinction between states
  - Random
  - BFS and Uniform Cost
  - DFS and Iterative Deepening
  - Backtracking
  - Bidirectional
- Informed - heuristics to help distinguish between states
  - Greedy best-first
  - Hillclimbing and Simulated Annealing
  - Beam search
  - A* and IDA*

Heuristics?

Common sense is as good as a mathematical proof.

Statistical proofs are hard to get and largely irrelevant.

Brute force (search) is great, but intelligence is faster.

Finding a relevant heuristic

A heuristic should **direct** the search towards the goal.

Heuristic: function applied to any state, returns extreme and opposing values for **all** initial state(s) and for **at least one** final state(s).

h: S -> [min, max], h(IS) = min/max, h(FS) = max/min

An admissible heuristic never overestimates the distance between a state and the goal.

Lets try finding a heuristic

For Hanoi Towers:

- Number of pieces on the goal tower
- If starting tower is 1 and goal is n, sum of all values in a state
- ?

For Sliding Puzzle:

- Number of pieces placed correctly

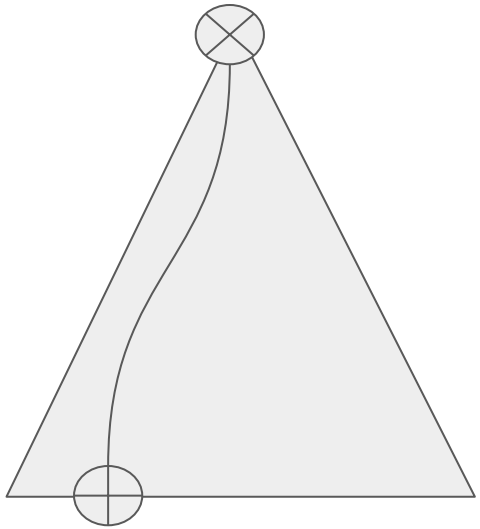
For chess:

- How many pieces I have - how many pieces has the opponent?
- A maximal value - number of moves available to the opponent?

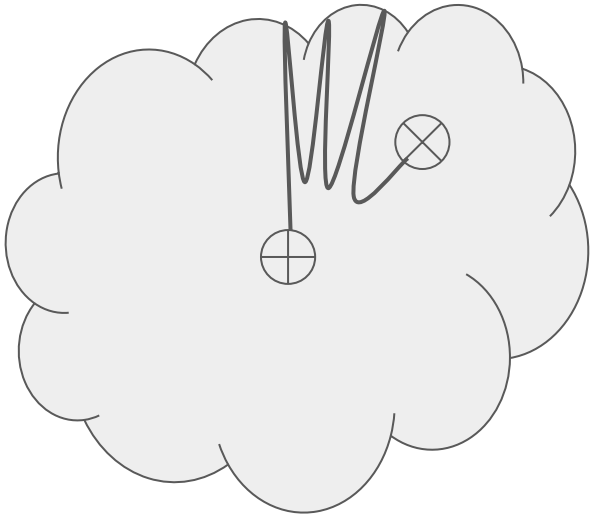
h should always be relatively easy to compute, good enough is enough

Three way of looking at solutions

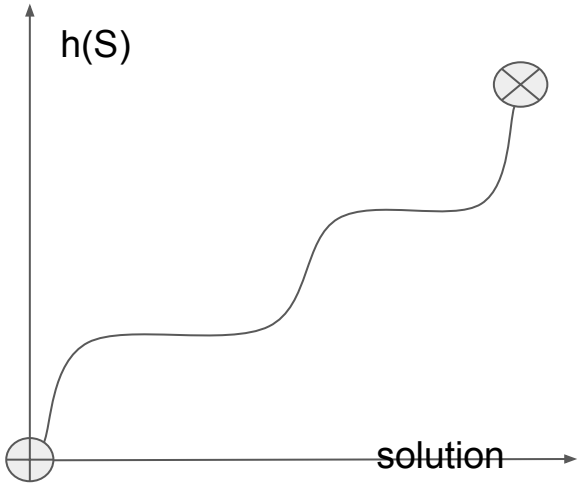
path in a graph



path in a space



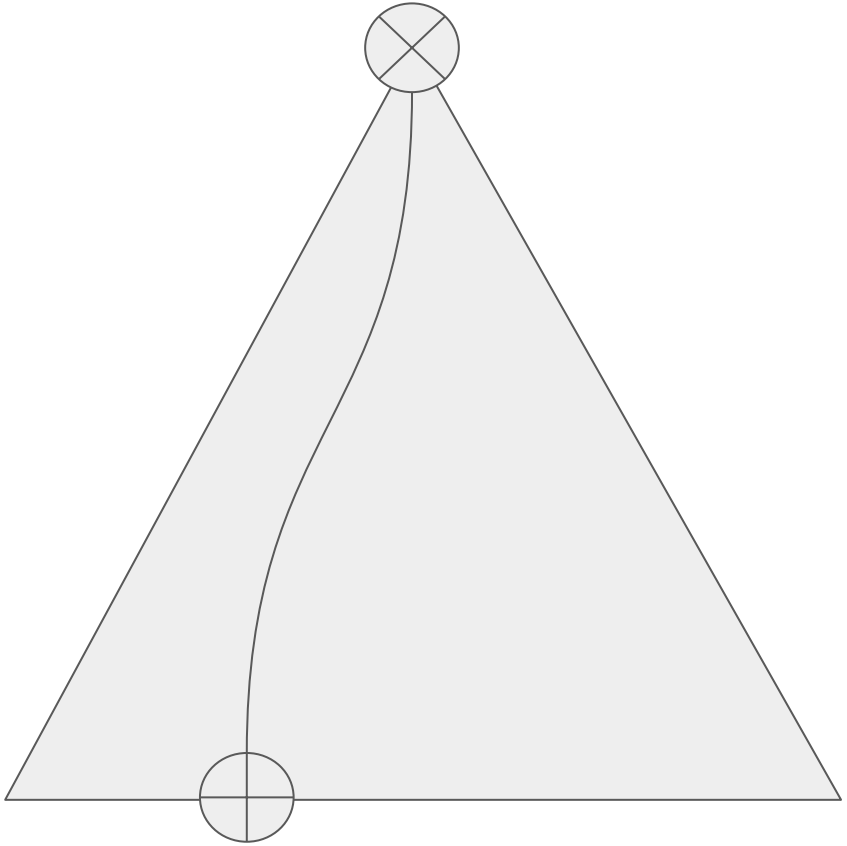
heuristic function graph



Greedy (best-first)

Evaluate **all** unexplored states accessible from the current state.

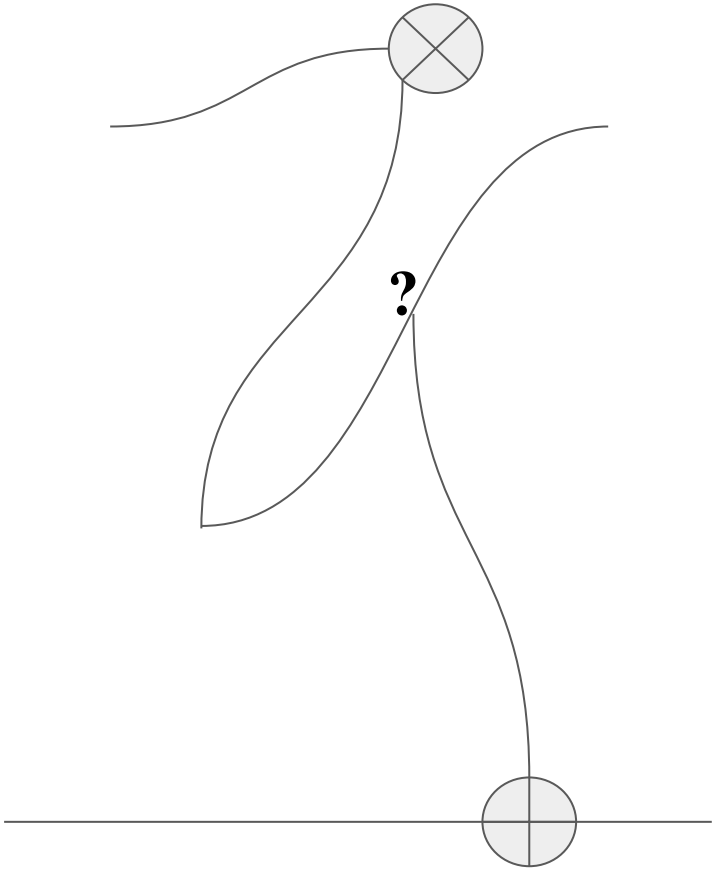
Select the unexplored state closer to the goal (best score according to a heuristic).



Greedy evaluation

Best case: gets you to the final state much quicker than DFS, doesn't guarantee optimal solution.

Worst case: DFS with bad choices.

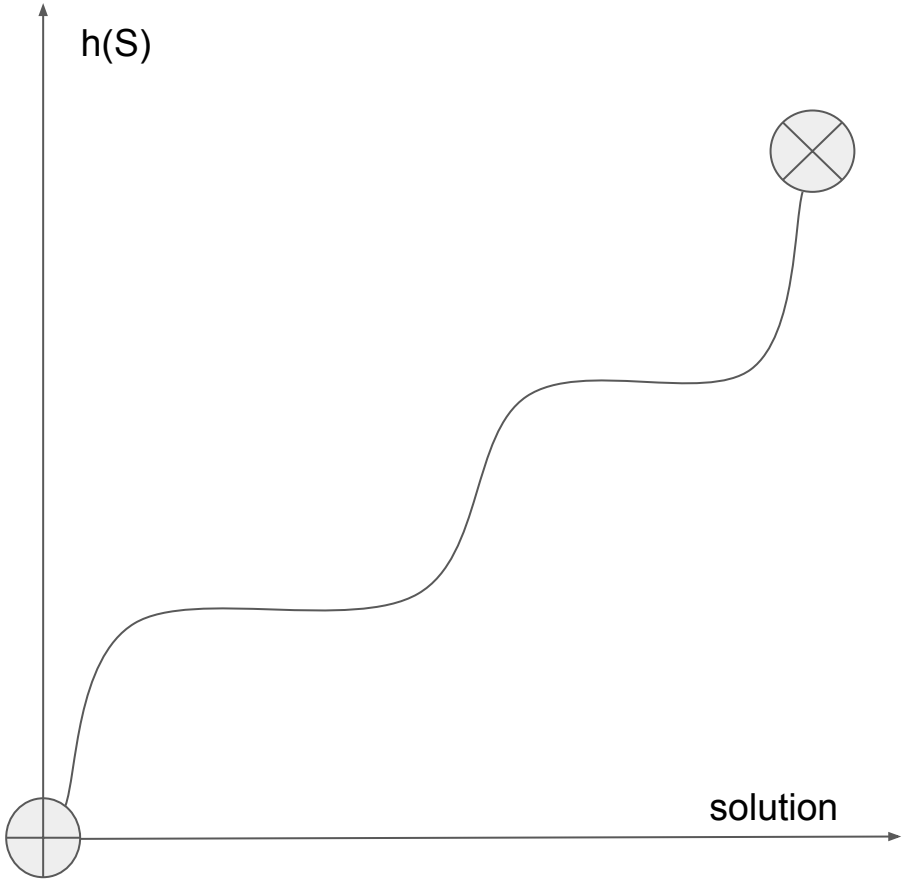


Hillclimbing

Select one accessible state which is at least as close to the final state as the current one.

h(NewState) ≥ h(CurrentState)

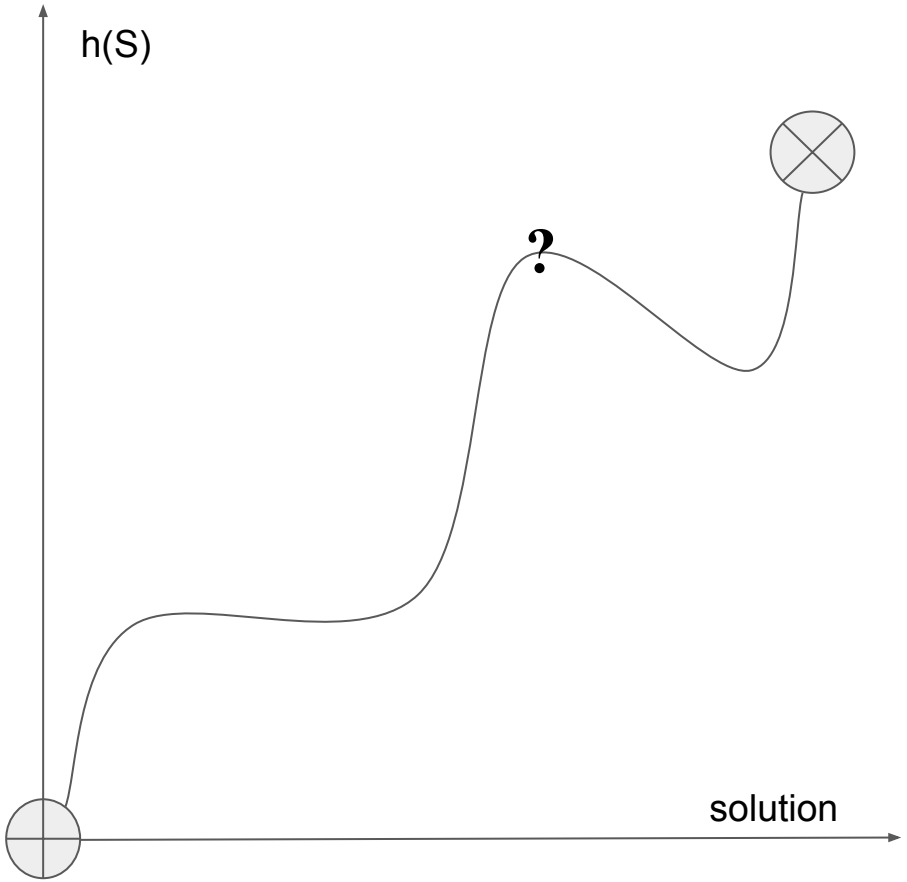
https://www.youtube.com/watch?v=j1H3jAAGlEA&list=PLUl4u3cNGP63gFHB6xb-kVBiQHYe_4hSi&index=4



Hillclimbing evaluation

Best case: fastest general strategy.

Worst case: fails to find a solution (local optimum).



Hillclimbing Selection Algorithm

```
eligible_neighbors = {};

For each state n reachable from current_state applying one valid transition
     If h(n)>=h(current_state) add n to eligible_neighbors;

Select next_state from eligible_neighbors;
```

Frequently used selections:

- First one found - most common
- Best one (greedy)
- All, in the order they were found, choosing next state available if stuck (hillclimbing-backtracking)

Simulated annealing

Hillclimbing, but allows choosing a state further away from the goal, with an ever decreasing probability the closer you get to the goal state.

No more local optimum.

Slower than hillclimbing.

http://www.cse.iitm.ac.in/~vplab/courses/optimization/SA_SEL_SLIDES.pdf

Beam search

BFS but only keeps in a sorted list best *k* visited states. The list should be re-sorted and extra states removed after every new explored state.

Explores all states in the sorted list until the final state is found - should be first in the list.

Does not guarantee the optimum solution!

A* and IDA*

A* is mostly Dijkstra combined with Greedy, IDA* is informed IDDFS

IDDFS explores nodes by distance from initial state d(S)

A* explores nodes by d(S) + h(S), where h(S) is the estimated distance to the goal state (obligatory an admissible heuristic)

Always chooses the unexplored state which is closest to both the initial state and the goal state.

https://www.redblobgames.com/pathfinding/a-star/introduction.html

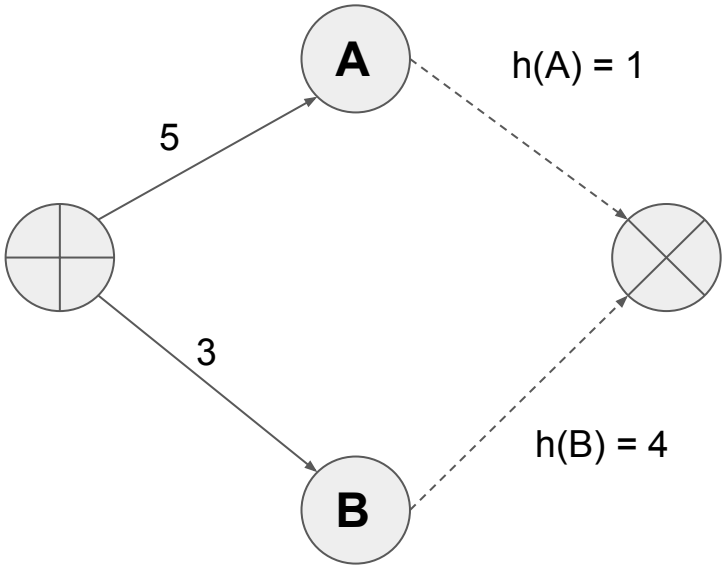
A* Algorithm

```
Current_state = Initial_state;
Best_score = maximum of h;
Sorted_queue = {All neighbours of Current_state, with computed h+1 scores, marked as
unexplored};

While the score of the first unexplored state S in Sorted_queue is lower than Best_score
     If S is a final state, Best_score = score of S;
     Mark S as explored;
     Add to Sorted_queue all its neighbours with computed h+d score and mark them
     unexplored;
     If a duplicate state appears in Sorted_queue, keep only the occurrence with lowest
     score and transfer explored status;
     Sort Sorted_queue;

The optimal path is recovered from the last state updating Best_score and looking for
explored neighbours with lower score up to the initial state.
```

A*

Always finds the optimum solution.

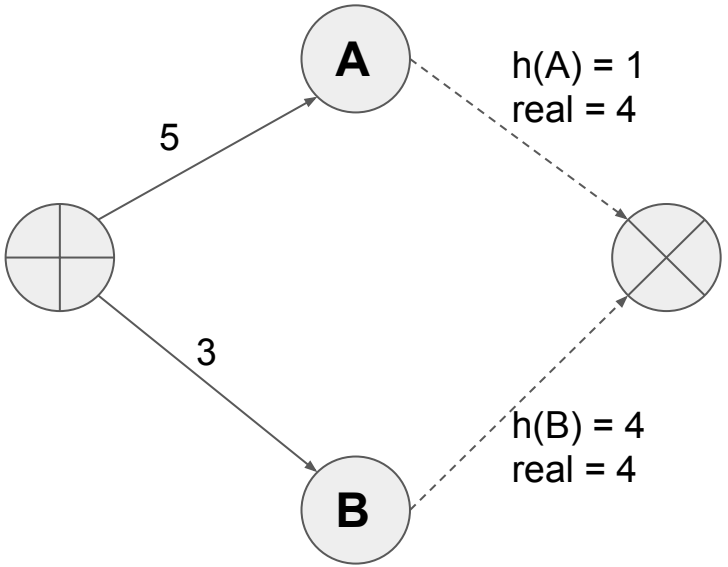
Really?



A*

Always finds the optimum solution.

Don’t stop when you find the goal state! Keep looking until all states with better scores have been explored.

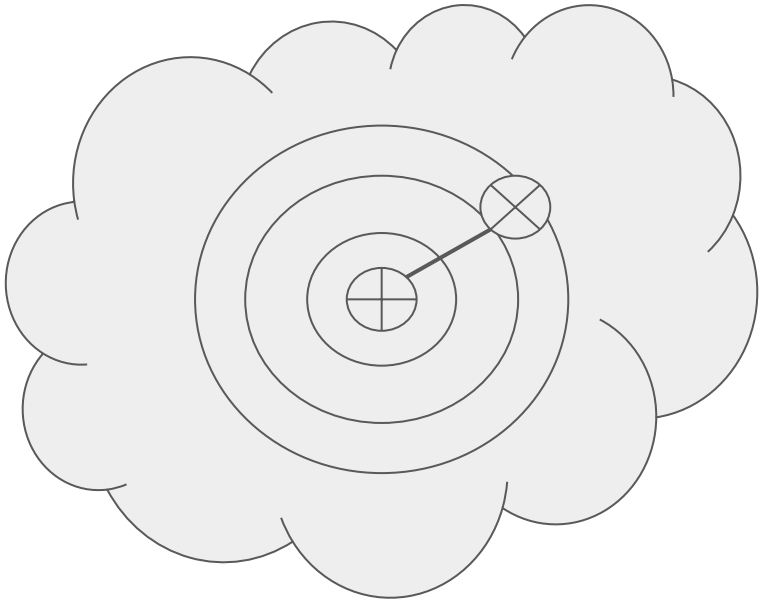


https://adrianstoll.com/a-star-pathfinding/

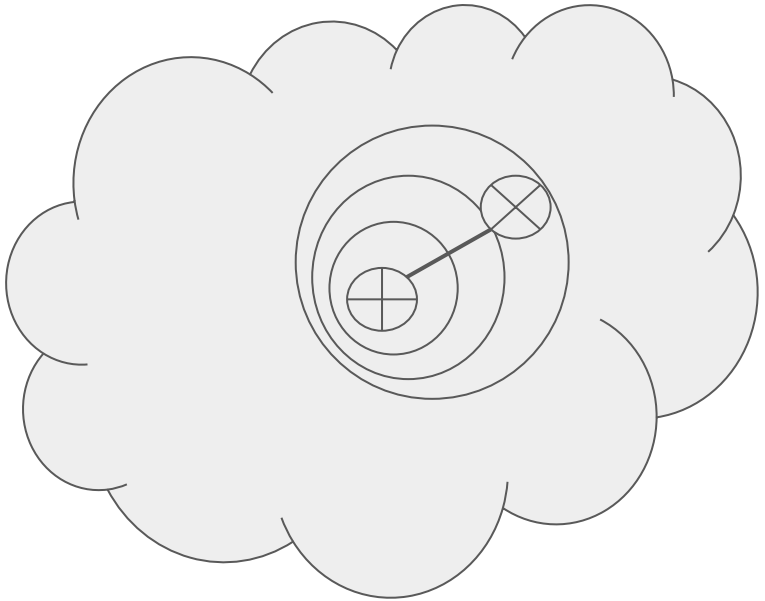
FACULTY OF
COMPUTER SCIENCE

IA@FII

BFS/IDDFS



A*



Optimizing A*

Consistent heuristic: $h(A) \leq h(A,B) + h(B)$ if B is reachable from A

Simplified Memory Bounded A*

- Prunes (removes) states for which the path was more costly than estimated, up to a set number of states (memory bounded)
- Remembers costs of reaching pruned states from each explored state
- If solution is not found, recovers pruned states and explores them

Choose a strategy

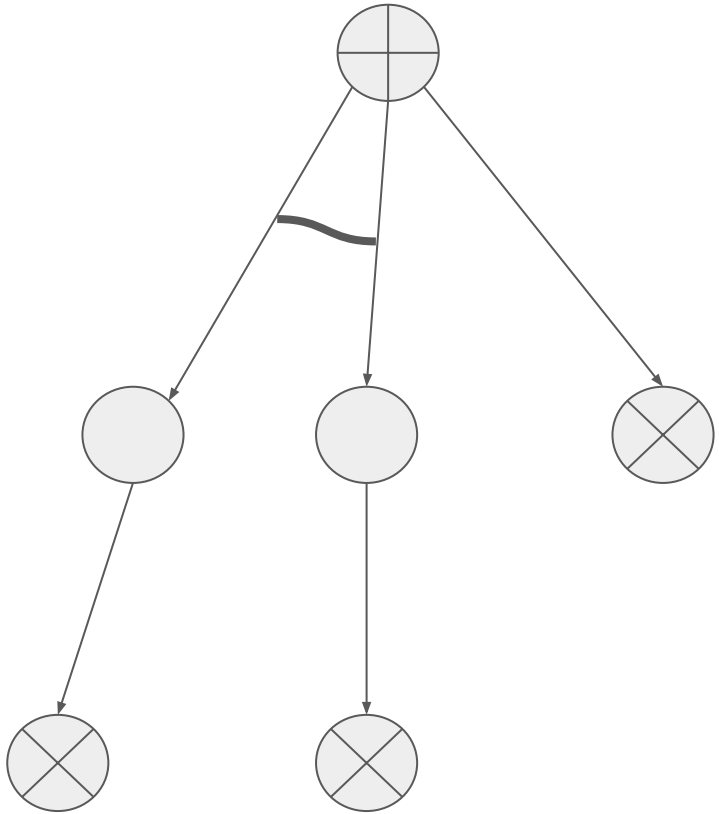
- Do you want a solution quickly?
- Do you want the best solution?
- Do you want all solutions?
- How much do you know about the problem?
- Are you sure the state representation is good enough?

Non-deterministic problem space

Stochastic: known possibilities and probabilities
Non-deterministic: known possibilities

Each transition has multiple possible outcomes

AND-OR trees



Partially observable problem space: you know what could happen after a transition, you don't know what will happen after a transition.

Unknowable problem space: you don’t know what could happen after a transition

Next week: Constraint satisfaction problems (Variable based models)

1								3
		7	2	6		4	8	
4			9	3	5			6
	3		4	8		2		
	4	1	6		9	3		
		6				8	9	
5	7	8		4				2
			3				7	
2								5

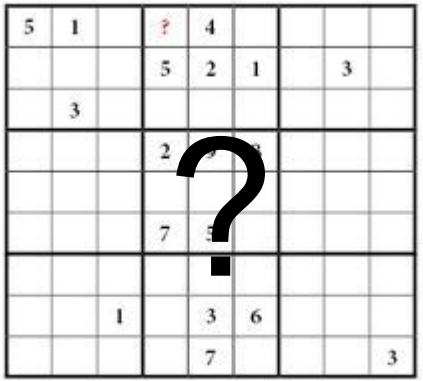
FACULTY OF
COMPUTER SCIENCE

IA@FII

Week 5: Games

Interactive decision problems (games):

- Solution is not built exclusively by your strategy
- Not all final states are the same



Hardest solvable problems in AI.
NLP problems hardest potentially solvable.