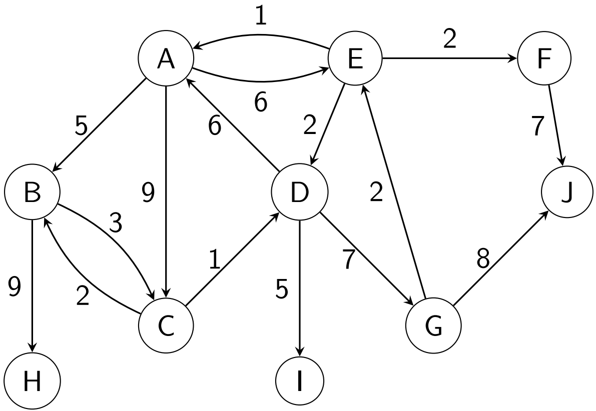
CS-541: Artificial Intelligence
Lecture 5

Abdul Rafae Khan
Department of Computer Science
Stevens Institute of Technology
akhan4@stevens.edu

February 22, 2022

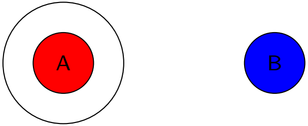
Uniform Cost Search

start state: A
goal state: H, I, J

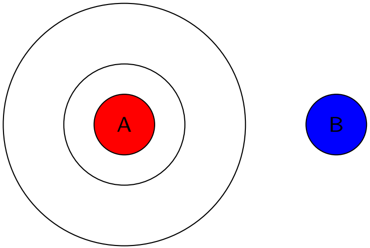


Uniform Cost Search

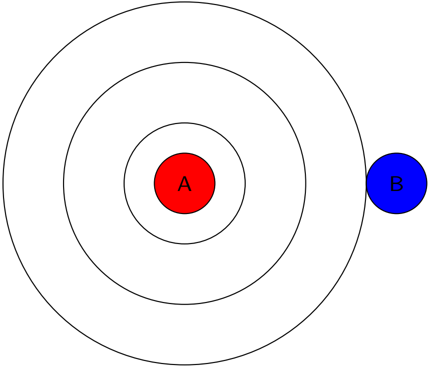
Uniform Cost Search



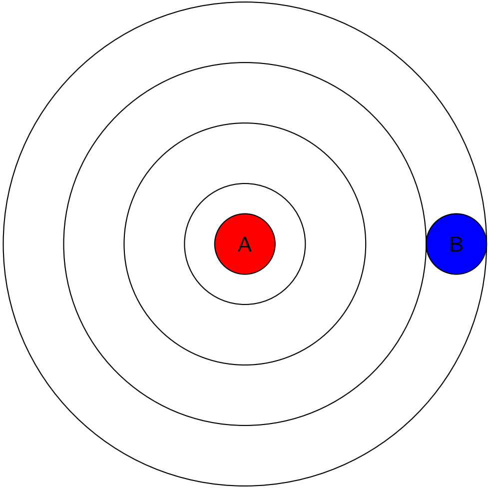
Uniform Cost Search



Uniform Cost Search

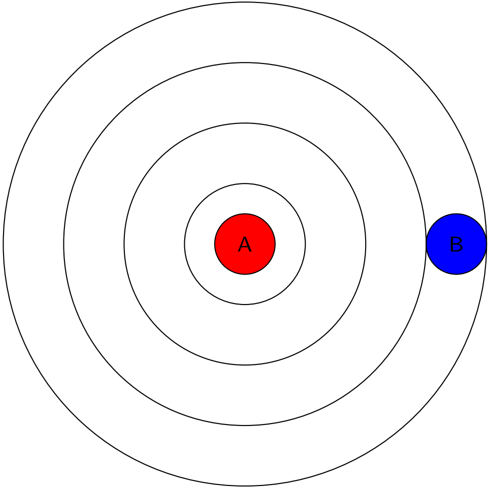


Uniform Cost Search

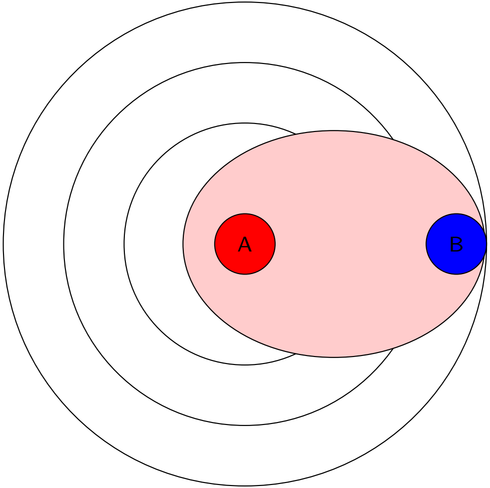


Uniform Cost Search

A lot of Wasted effort!



Informed Search

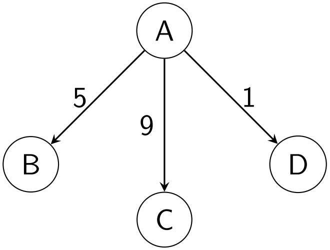


Informed Search

Uniform cost search:

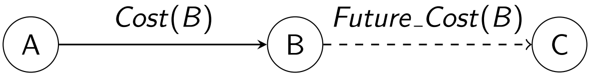
Search using: $\min_{a \in Actions(s)} Cost(s, a)$

At every state s, select the action a with the minimum cost



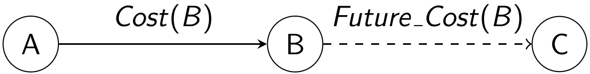
Informed Search

$Total\_Cost(goal) = Cost(s, a) + Future\_Cost(s)$
$Future\_Cost(s)$ is the cost to go from state s to state $goal$
If we know the $Future\_Cost(s)$, then our search would be better



Informed Search

Informed cost search:
Search using: $\min_{a \in Actions(s)} Cost(s, a) + Future\_Cost(s)$
At every state s, select the action a with the minimum cost



Informed Search

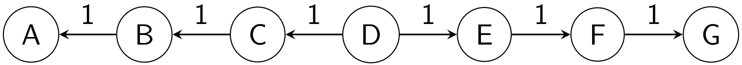
If we know $Future\_Cost(s)$, we know the solution!

Informed Search

Lets estimate the $Future\_Cost(s)$
$Cost'(s) = Cost(s) + h(Successor(s)) - h(s)$
$h(s)$ is an estimate of the cost from s to the *goal*

A* Search

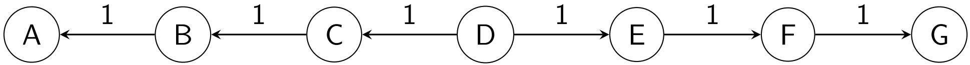
Start: D
Goal: G



Whiteboard

A* Search

Start: D
Goal: G



Node	A	B	C	D	E	F	G
$h(s)$	6	5	4	3	2	1	0

A* Search

Start: D

Goal: G

$$\begin{tikzcd}
A & B \arrow[l, "\cancel{1}\ 2"'] & C \arrow[l, "\cancel{1}\ 2"'] & D \arrow[l, "\cancel{1}\ 2"'] \arrow[r, "\cancel{1}\ 0"] & E \arrow[r, "\cancel{1}\ 0"] & F \arrow[r, "\cancel{1}\ 0"] & G
\end{tikzcd}$$

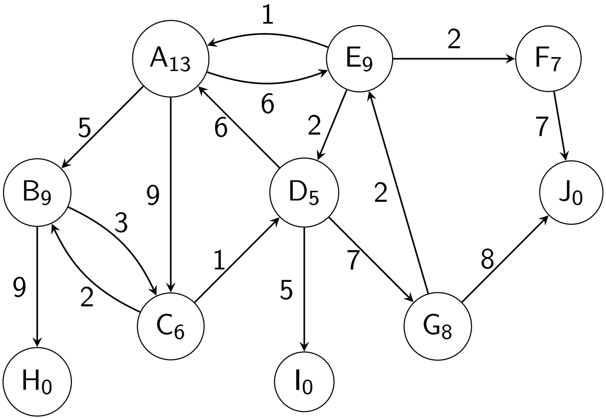
Node	A	B	C	D	E	F	G
$h(s)$	6	5	4	3	2	1	0

$$Cost'(C) = Cost(C) + h(C) - h(D) = 1 + 4 - 3 = 2$$

Whiteboard

A* Search

start state: A
goal state: H, I, J



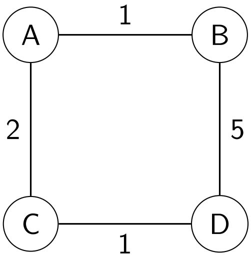
Whiteboard

Tree Search

Can any heuristic work?

Tree Search

Can any heuristic work?
start state: A
goal state: D



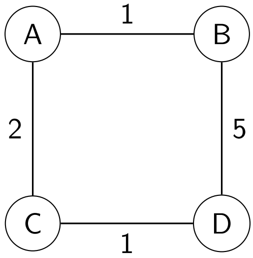
Whiteboard

Tree Search

Can any heuristic work?

start state: A

goal state: D



Node	A	B	C	D
$h(s)$	0	0	100	0

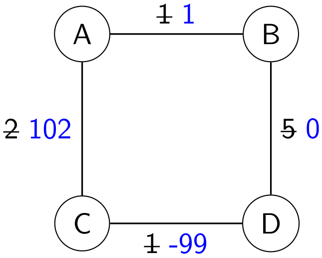
Whiteboard

Tree Search

Can any heuristic work?

start state: A

goal state: D



Node	A	B	C	D
$h(s)$	1	1	100	0

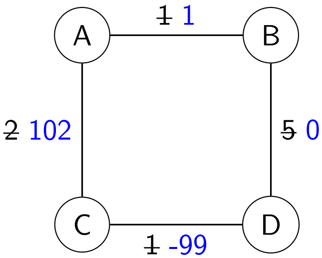
Whiteboard

Tree Search

Can any heuristic work?

start state: A

goal state: D



The algorithm does not work with **negative edge costs**

Whiteboard

Conditions for heuristics

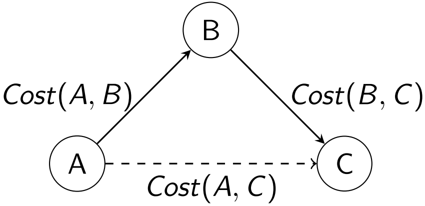
Consistency

1. $h(successor(s)) - h(s) \leq Cost(s, successor(s))$

Conditions for heuristics

Consistency

1. $h(successor(s)) - h(s) \leq Cost(s, successor(s))$



$h(A) - h(C) \leq Cost(A, C)$
or $h(A) \leq Cost(A, C) + h(C)$

Conditions for heuristics

Consistency

1. $h(successor(s)) - h(s) \leq Cost(s, successor(s))$
2. $h(goal) = 0$

Conditions for heuristics

Consistency

1. $h(successor(s)) - h(s) \leq Cost(s, successor(s))$
2. $h(goal) = 0$

Admissibility

1. $h(s) \leq Cost(s, goal)$

Conditions for heuristics

Consistency

1. $h(successor(s)) - h(s) \leq Cost(s, successor(s))$
2. $Future\_Cost(goal) = 0$

Admissibility

1. $h(s) \leq Cost(s, goal)$, heuristic cost is less than actual cost

$h(A) \leq Cost(A, C) + h(C) \implies g(A) + h(A) \leq g(A) + Cost(A, C) + h(C)$

Conditions for heuristics

Consistency

1. $h(successor(s)) - h(s) \leq Cost(s, successor(s))$
2. $Future\_Cost(goal) = 0$

Admissibility

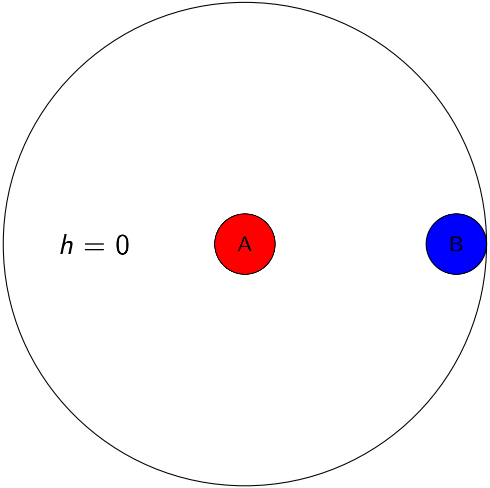
1. $h(s) \leq Cost(s, goal)$, heuristic cost is less than actual cost

$$h(A) \leq Cost(A, C) + h(C) \implies g(A) + h(A) \leq g(A) + Cost(A, C) + h(C)$$

$$h(s) \text{ is consistent } \implies h(s) \text{ is admissible}$$

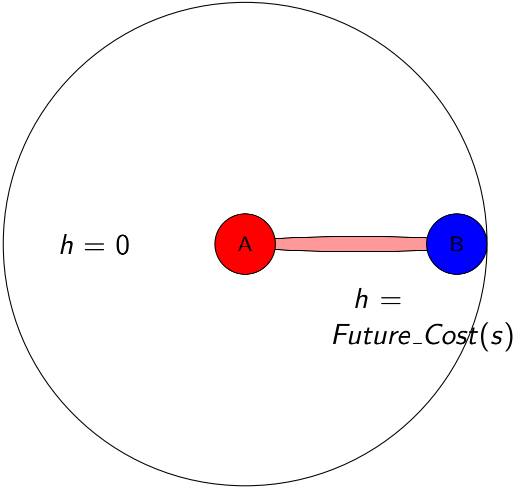
Informed Search

If $h(s) = 0$, A^* is the same as UCS



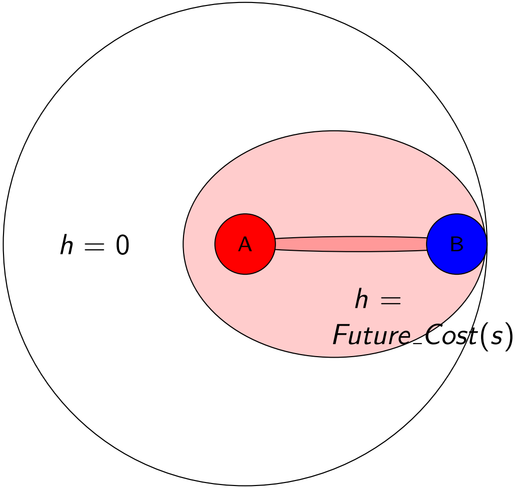
Informed Search

If $h(s) = Future\_Cost(s)$, the A^* only explores the minimum cost nodes



Informed Search

Usually $h(s)$ is in between



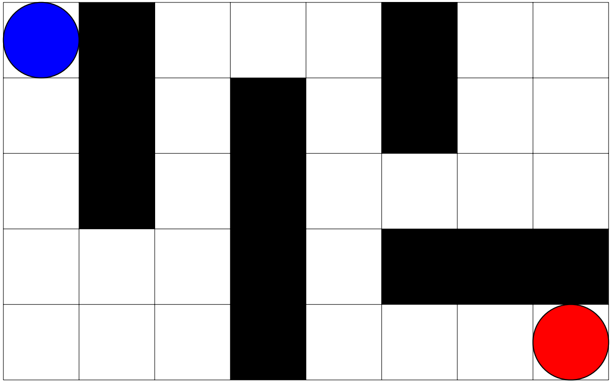
Heuristics

How to get heuristic values?

Relaxations

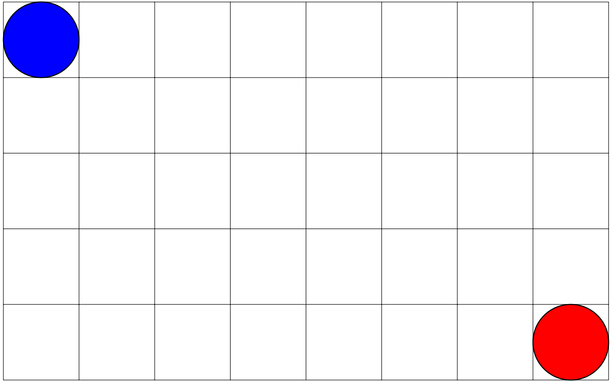
How to get heuristic values? Make the problem easy!

Relaxations

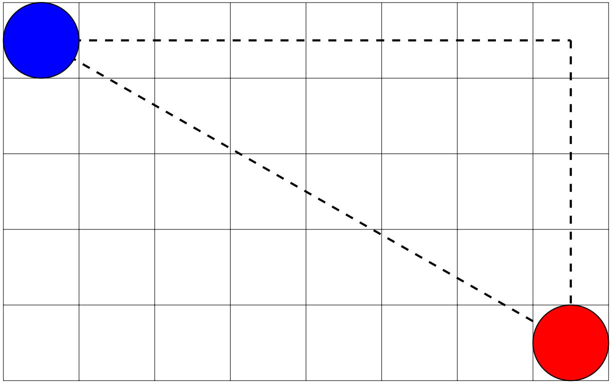


Relaxations

Drop all the edges!

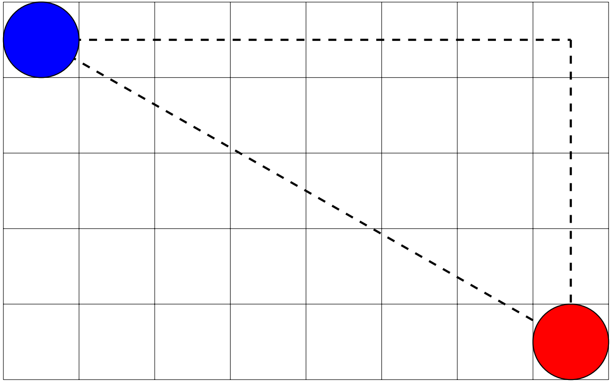


Relaxations



- $h(start) = Manhattan(start, goal)$
- $h(start) = Euclidean(start, goal)$

Relaxations



- $Cost(start, right) = \infty$
- $Cost'(start, right) = \mathbb{R}^+$

Relaxations

Goat & Cabbage/Goat & Wolf can be left alone

Relaxations

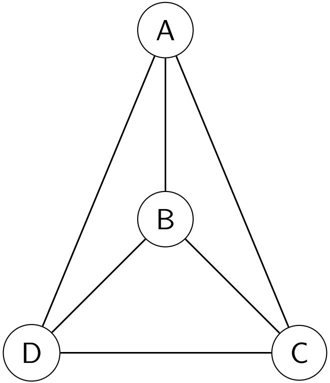
Goat & Cabbage/Goat & Wolf can be left alone
Similarly for other problems!

Local Search

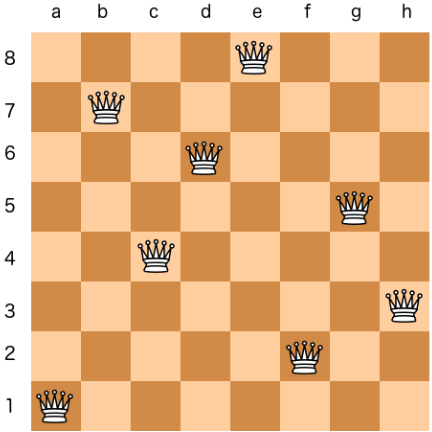
Sometimes the path is not important
The goal is the solution we want to achieve!

Traveling Salesman Problem

Start at state A and reach state C
while visiting all the states exactly once



8-Queen Problem



Place 8 queens on a chessboard such that no queen is being attacked

Local Search

Sometimes the path is not important
The goal is the solution we want to achieve!

Do not require a lot of memory
Work with continuous space

Local Search

Sometimes the path is not important
The goal is the solution we want to achieve!

Do not require a lot of memory
Work with continuous space

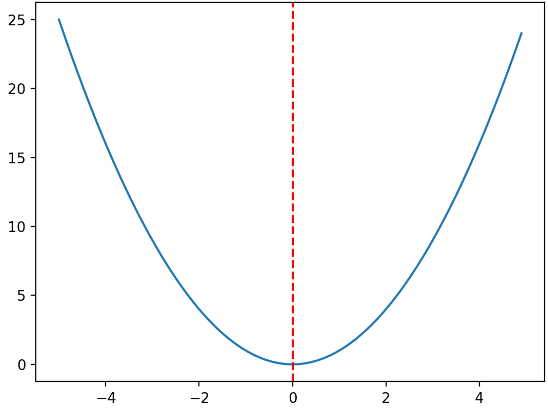
You can think of it as optimizing a criterion
You can iteratively try to improve the current state

Hill Climbing

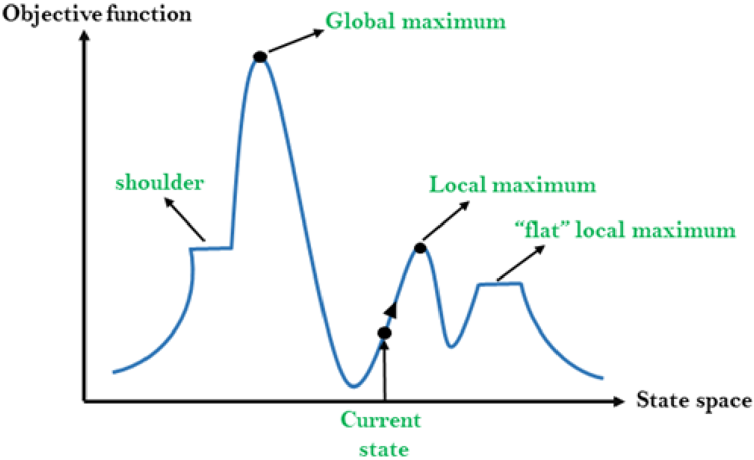
Start at any state
Repeatedly move to the best neighboring state
If no neighboring state is better, stop

Local vs Global Maxima/Minima

Maximize the benefit or minimize the cost

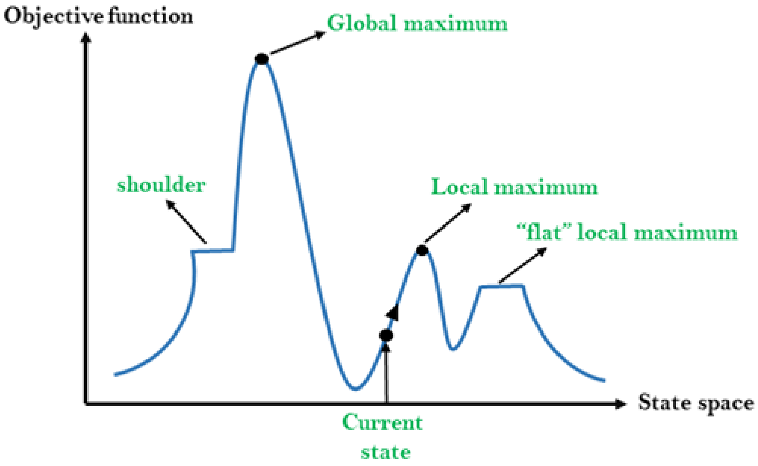


Local vs Global Maxima/Minima



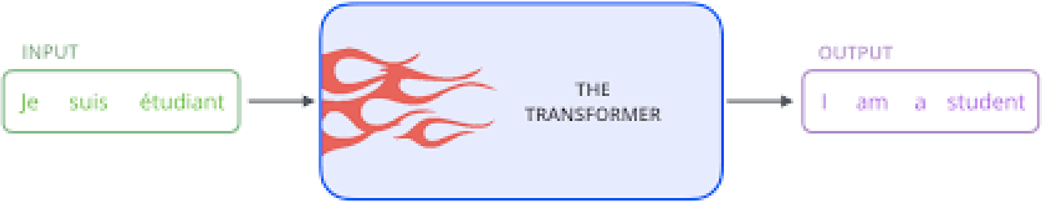
Local vs Global Maxima/Minima

Restart hill climbing from a random state



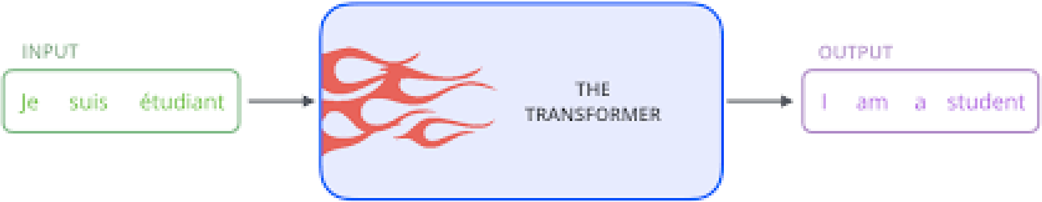
Beam Search

Lets say we have a machine translation model already trained
Translation output is one word at a time
We want to use it to output the best translation for any given input sentence



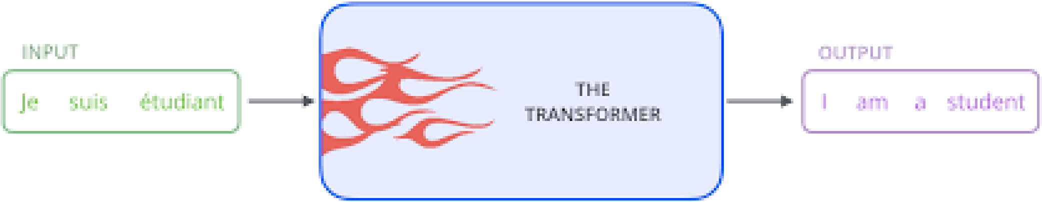
Beam Search

Given the input sentence (french) and the previous output word (English), the model outputs the next word



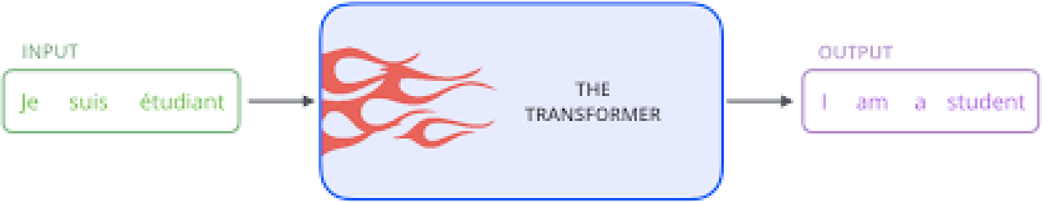
Beam Search

If we have V words in the vocabulary and maximum sentence length is L
Exhaustive search will be V^L
If $vocab = 1000$ and $max\_length = 10$, then 10^{40} possible choices to select from



Beam Search

A simpler option is to select the best at each position



Beam Search

Typically greedy selection for the word at each position

Position	1	2	3	4
A	0.5	0.1	0.2	0.1
B	0.2	**0.4**	0.2	0.2
C	0.2	0.3	0.4	0.1
<END>	0.1	0.2	0.1	0.6

Output probability $= 0.5 \times 0.4 \times 0.4 \times 0.6 = 0.048$

Beam Search

It can happen that this is not the optimal
Selecting 2*nd* best at position 2 gives better total probability

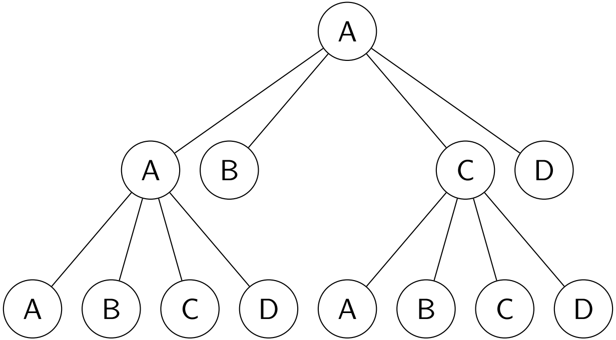
Position	1	2	3	4
A	0.5	0.1	0.1	0.1
B	0.2	0.4	0.6	0.2
C	0.2	**0.3**	0.2	0.1
<END>	0.1	0.2	0.1	0.6

Output probability $= 0.5 \times 0.3 \times 0.6 \times 0.6 = 0.054$

Beam Search

1*st* step:
Select the K maximum probability words
For each subsequent step:
Fix the previous selections and generate K possibilities and select the overall K maximum

Beam Search



Whiteboard

Games

All previous algorithms work for single agent only
What if we have more than one agent?

Games

All previous algorithms work for single agent only
What if we have more than one agent?

- Tic-Tac-Toe
- Pacman
- Chess

Games

Assumptions:

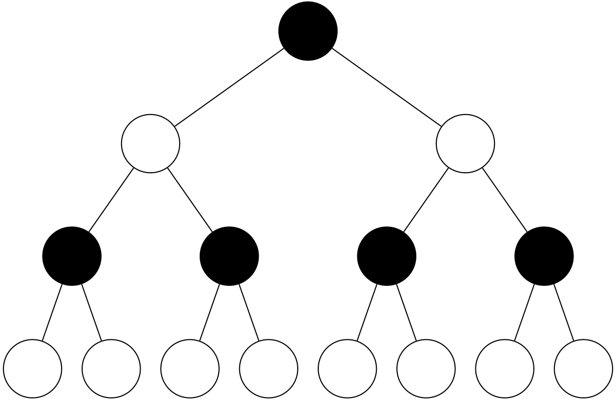
- States and transitions are deterministic
- Environment is fully observable
- 2 agents with turn-taking
- Zero-sum game: In which result is an advantage for one side and a loss for the other (One winner & One loser)

Algorithm to compute the strategy for to recommend every possible move for

Tree Search

Each node represents a turn by a player
One player tries to maximize a utility and the other tries to minimize

Minmax Search



Search Tree

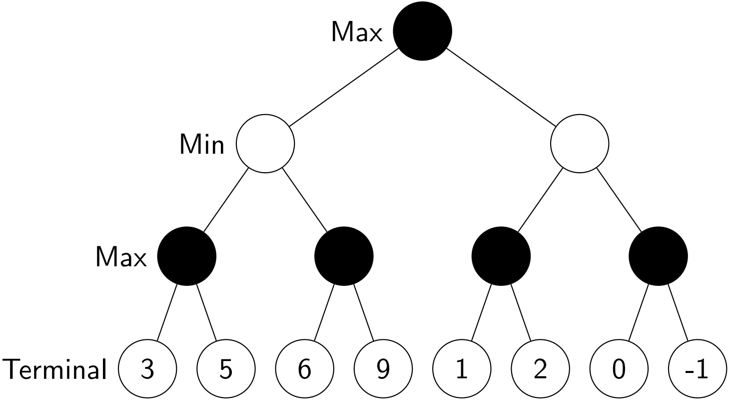
Each node represents a turn by a player
One player tries to maximize a utility and the other tries to minimize
Leaf nodes represent some utility value
Root to leaf is a path for the outcome of the game

Each player's turn is called a ply
One turn is the play by both the players

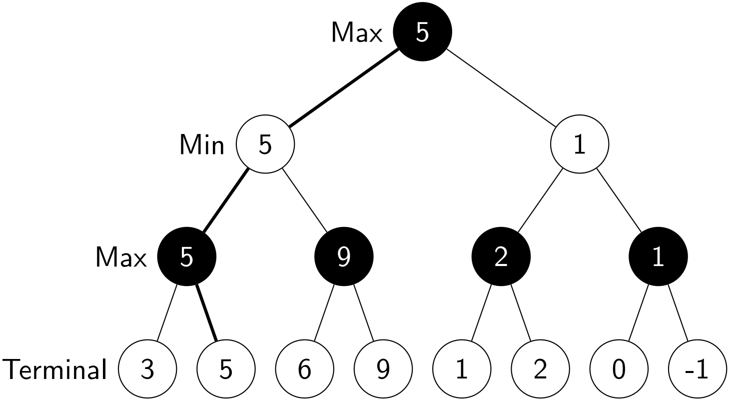
Search Tree

Leaf nodes can represent a utility value
What can be possible values for
Tic-Tac-Toe
Chess
Pacman

Search Tree



Minmax Search

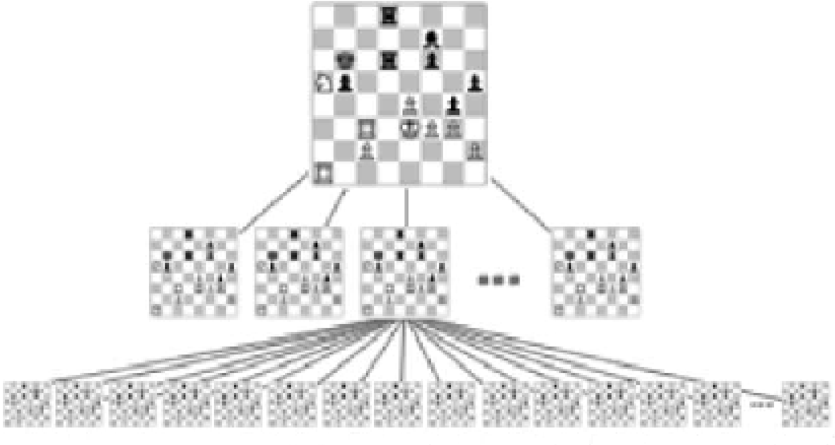


Whiteboard

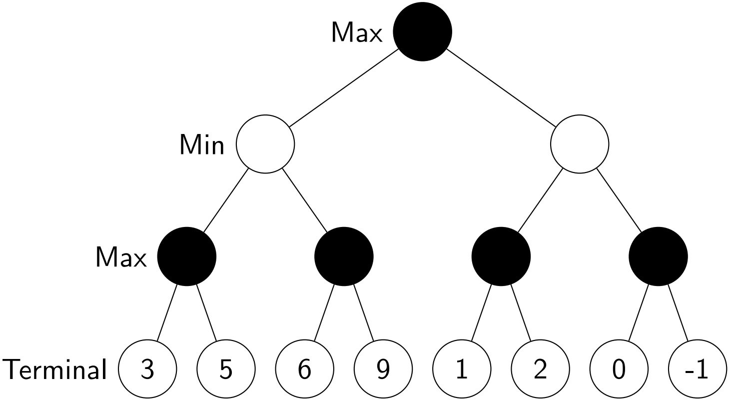
Alpha-Beta Pruning

The complete tree can be very big

E.g. Just the first 5 turns (10 plies) of chess generate **69,352,859,712,417** possibilities

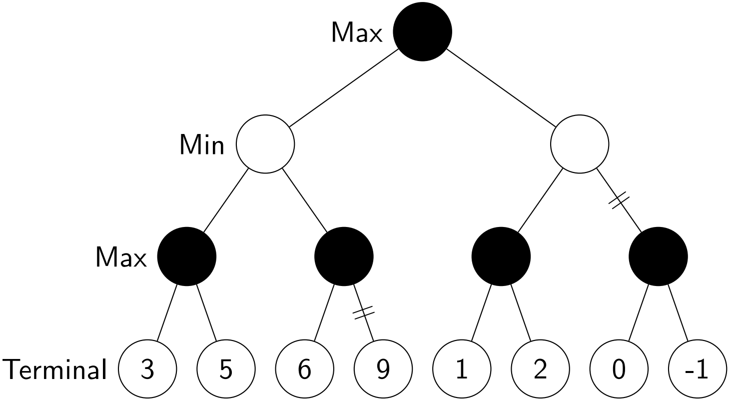


Alpha-Beta Pruning



Whiteboard

Alpha-Beta Pruning



Whiteboard

Alpha Beta Pruning

```
def max_value(s,α,β):
  if leaf_node(s):
    return value(s)
  v=−∞
  for each u in successor(s):
    v′ = max(v, min_value(u,α,β))
    if v′ ≥ β:
      return v
    α = min(v,\alpha)
  return v
```

```
def min_value(s,α,β):
  if leaf_node(s):
    return value(s)
  v=+∞
  for each u in successor(s):
    v′ = min(v, max_value(u,α,β))
    if v′ ≤ α:
      return v
    β = min(v,β)
  return v
```

Recap

Informed Search

- A^* Search
- Heuristics

Local Search

- Hill Climbing
- Beam Search

Zero-Sum Games

- Minimax Search
- Alpha-Beta Pruning

References

Stuart Russell and Xiaodong Song (2021)
CS 188 — Introduction to Artificial Intelligence
University of California, Berkeley

Chelsea Finn and Nima Anari (2021)
CS221 — Artificial Intelligence: Principles and Techniques
Stanford University

The End