UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Adversarial Search

Alfons Juan
Jorge Civera

DSIC

Departament de Sistemes Informàtics i Computació

Objectives

- To know the basics of adversarial search.
- To apply the ***minimax*** algorithm and ***alpha-beta*** pruning.

Contents

1 Adversarial search

Adversarial search consists in finding the best move in games being:

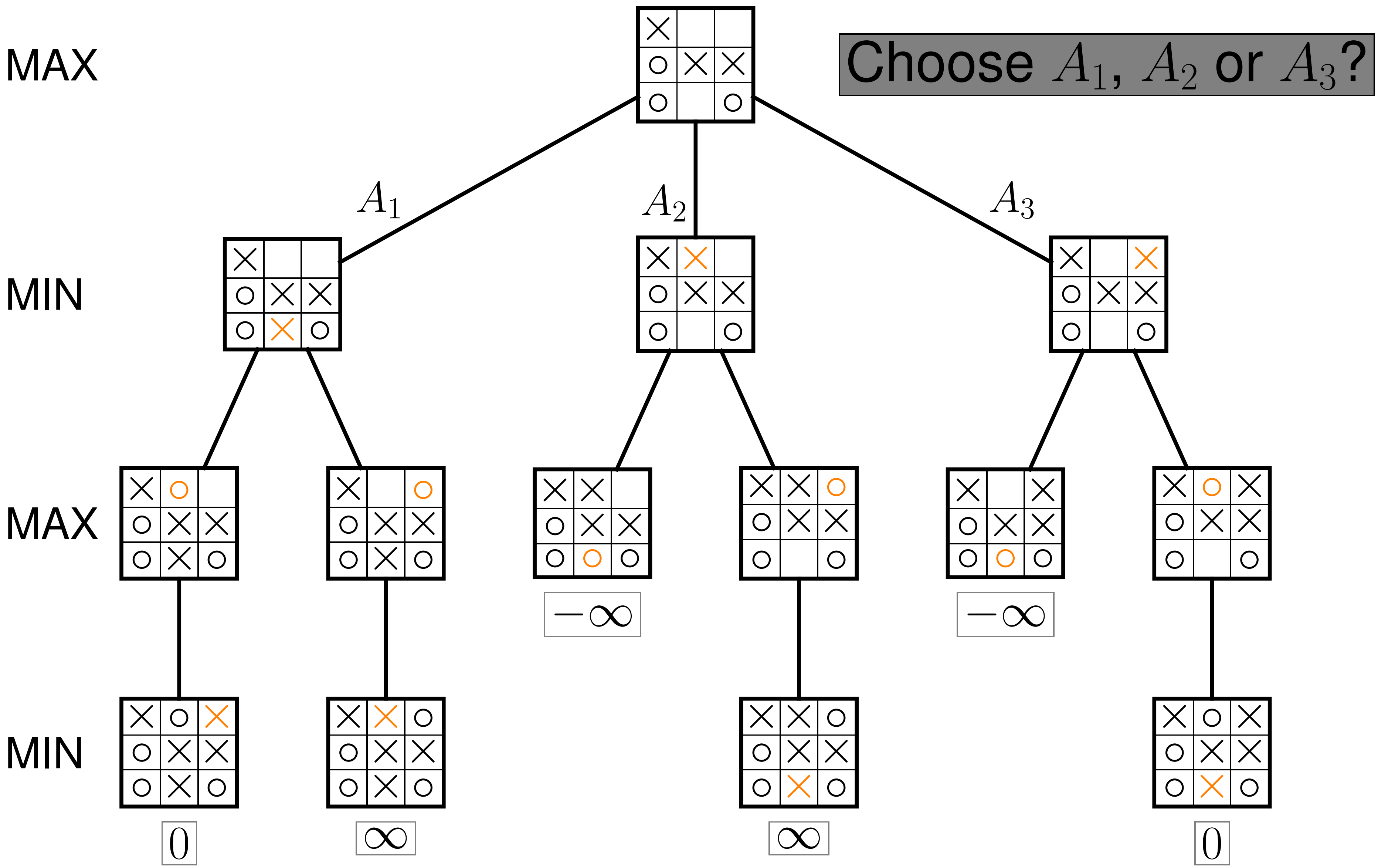
- ***Deterministic*** i.e. luck does not play a role
- ***Two-player*** MAX (the system) and MIN (the opponent)
- ***Turn-taking*** MAX starts and decides its move
- ***Perfect info*** states and rules are known (i.e. chess)
- ***Zero-sum*** MAX/MIN utilities balanced at the end of the game

Basic elements:

- ***Initial state s_0:*** from which MAX chooses the best move.
- ***Actions(s):*** set of legal moves from state s.
- ***Terminal(s):*** true if the game in s is over and false otherwise.
- ***Utility(s):*** utility for MAX of the terminal state s.

Goal: choose a move leading to a state of maximum utility

Example: Choose a move in tic-tac-toe



2 Minimax algorithm and alpha-beta pruning

Minimax value, decision and algorithm:

- ***Minimax value*** of a state/node: utility (for MAX) of the terminal state that we reach if both players play optimally.
- ***Minimax decision:*** Choose the move with highest minimax value
- ***Minimax algorithm:*** Computation of minimax decision based on (bounded) depth-first adversarial search.

Basic minimax algorithm

mm(s**,** d**,** max**)** // state, depth, max="Does MAX move?"

  if s is terminal: **return** utility for s

  if $d = 0$: **return** heuristic value for s

  // if max, return maximum minimax value from children

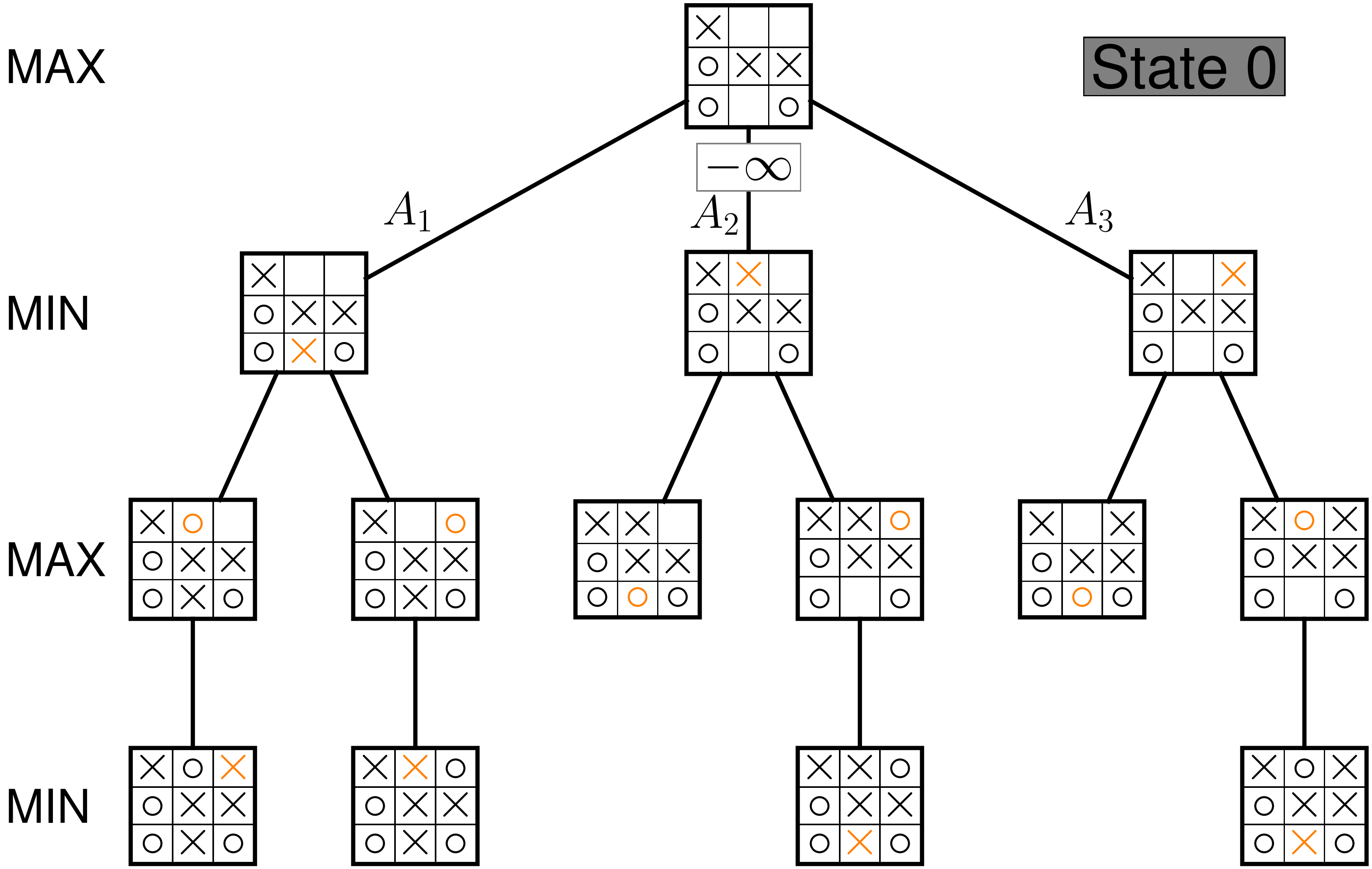
  if max: $v = -\infty$; $\forall\, n \in$ succ(s): $v = \max(v, \textbf{mm}(n, d-1, \text{FALSE}))$

  // if min, return minimum minimax value from children

  else: $v = \infty$; $\forall\, n \in$ succ(s): $v = \min(v, \textbf{mm}(n, d-1, \text{TRUE}))$

  return v

Solution for minimax example



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Minimax algorithm and alpha-beta pruning

mm(s, d, max) // state, depth, max="Does MAX move?"

  if s is terminal: **return** utility for s

  if $d = 0$: **return** heuristic value for s

  if max: $v = -\infty$; $\forall\, n \in$ succ(s): $v = \max(v, \mathbf{mm}(n, d-1, \text{FALSE}))$

  else: $v = \infty$; $\forall\, n \in$ succ(s): $v = \min(v, \mathbf{mm}(n, d-1, \text{TRUE}))$

  return v

α-β(s, d, α, β, max)

  if s is terminal: **return** utility for s

  if $d = 0$: **return** heuristic value for s

  if max: $v = -\infty$

    $\forall\, n \in$ succ(s)

      $v = \max(v, \alpha\text{-}\beta(n, d-1, \alpha, \beta, \text{FALSE}))$

      $\alpha = \max(\alpha, v)$; **if** $\beta \leq \alpha$: **break** // β cut

  else: $v = \infty$

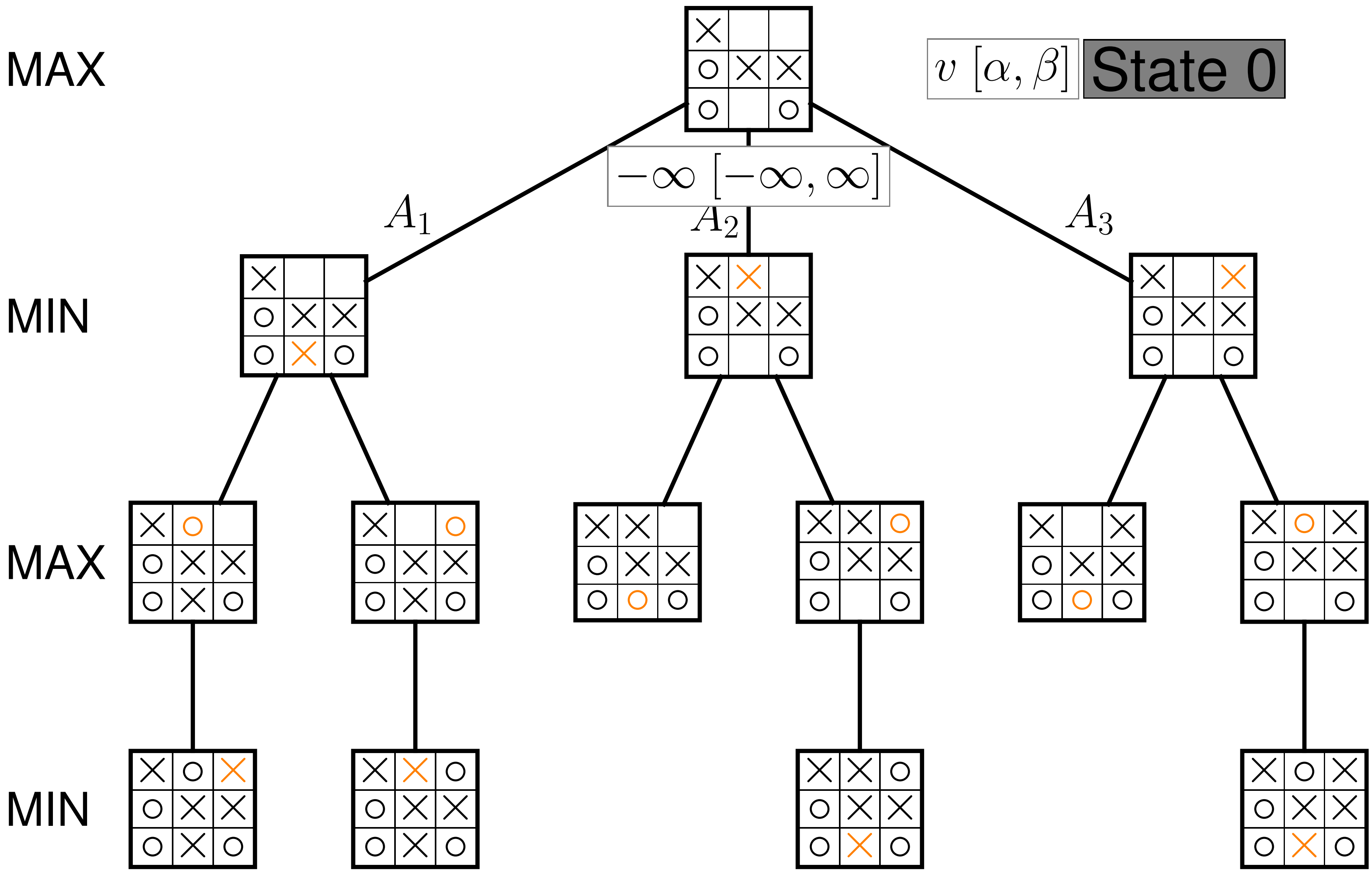
    $\forall\, n \in$ succ(s)

      $v = \min(v, \alpha\text{-}\beta(n, d-1, \alpha, \beta, \text{TRUE}))$

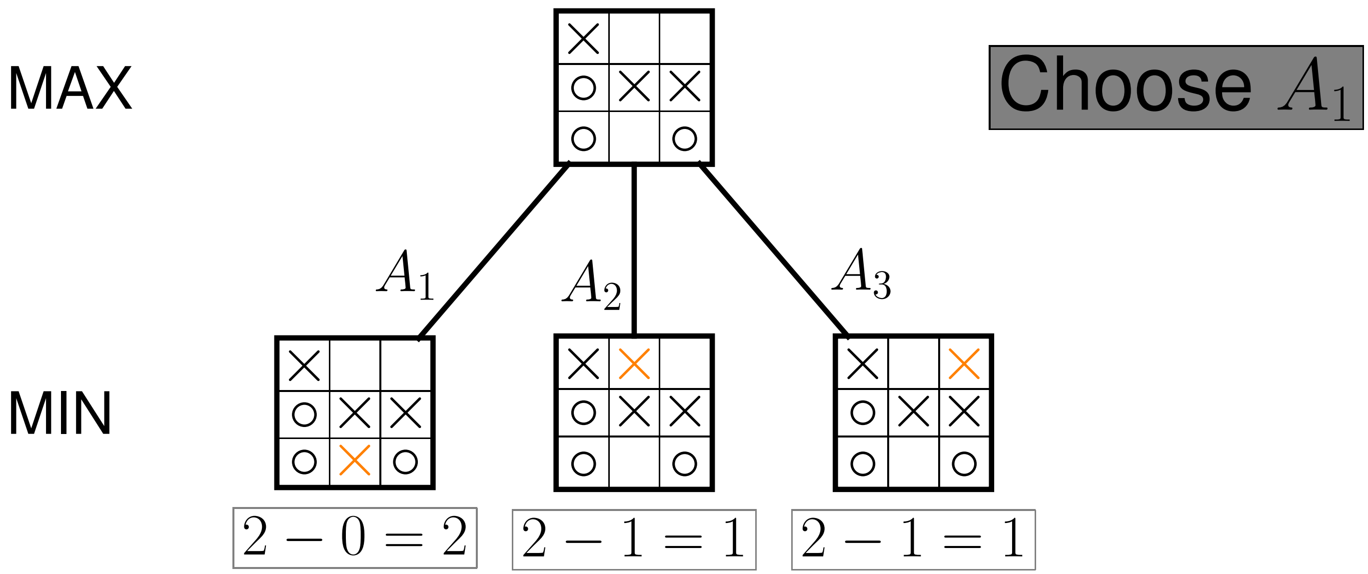
      $\beta = \min(\beta, v)$; **if** $\beta \leq \alpha$: **break** // α cut

  return v

Solution for example of alpha-beta pruning



Solution for example max depth *d*=1 and heuristic



Heuristic function:

$$h(n,j) = \text{open(n,MAX)} - \text{open(n,MIN)}$$

where

Open(n,j)="After placing player j their mark in all empty squares, number of their winning combinations"

Conclusions

- We have studied the basics of adversarial search.
- We have applied the ***minimax*** algorithm and ***alpha-beta*** pruning.
- See [1, Chapter 5] for more details.

References

[1] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, third edition, 2010.