Wang Xiyu

National University of Singapore
School of Computing
CS2109S: Introduction to AI and Machine Learning
Semester 2, 2024/2025

**Tutorial 3**
**Adversarial Search and Local Search**

## Summary of Key Concepts

In this tutorial, we will discuss and explore the following key learning points/lessons.

1. Adversarial Search
   (a) Minimax Algorithm
   (b) $\alpha$-$\beta$ Pruning
2. Local Search

## A Adversarial Search in Tic-Tac-Toe

1. Consider the game tree for Tic-Tac-Toe shown in Figure 1. The Tic-Tac-Toe search space can actually be reduced by means of symmetry. This is done by eliminating those states which become identical with an earlier state after a symmetry operation (e.g. rotation). Figure 2 explores the possible next moves after the "x" player has made the first move in the center, and the "o" player made the second move in the top-middle. Assume that the following heuristic evaluation function is used at each node $n$:

$$Eval(n) = X(n) - O(n)$$

where $X(n)$ is the number of possible winning lines for the "x" player while $O(n)$ is the number of possible winning lines for the "o" player. Hence, the "x" player will look to maximize $Eval(n)$, whereas the "o" player will look to minimize it.

   (a) Assume that the "x" player places his first move in the centre and the opponent has responded with an "o". Compute the evaluation function for each leaf node and determine the next move of the "x" player in Figure 2.

   A dashed arrow indicates a "x" player move ($Max$ player); a full arrow indicates a "o" player move ($Min$ player).

   A few heuristic calculations have been done for you as examples.

# A

## a

$$Eval(n) = X(n) - O(n)$$

Winning line here means potential wining combinations for one side:

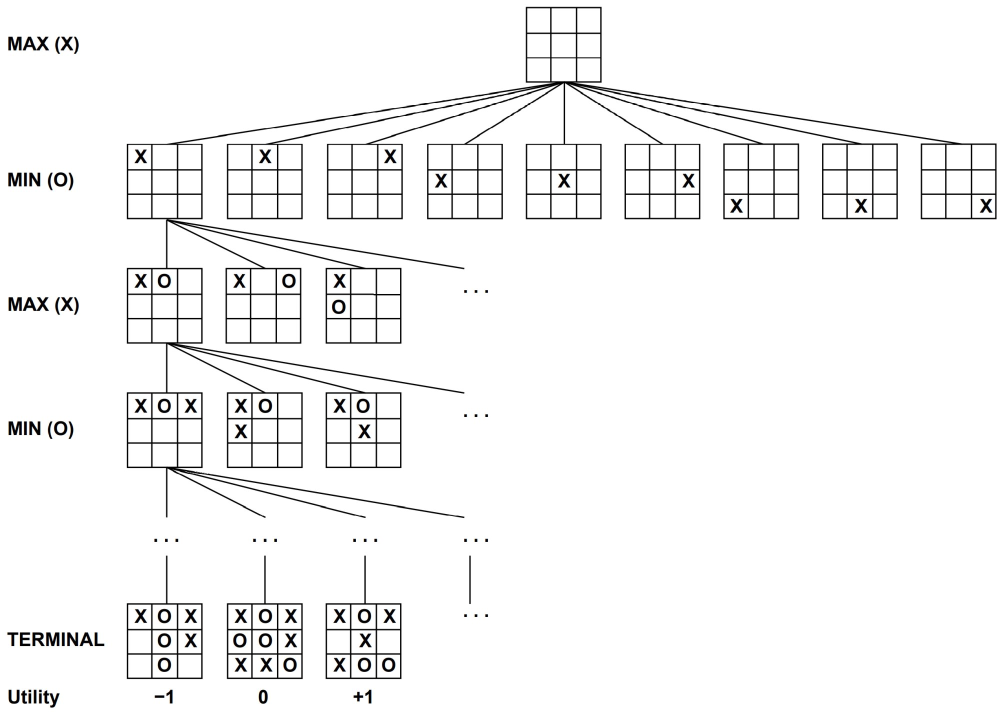



Figure 1: Search space for Tic-Tac-Toe.

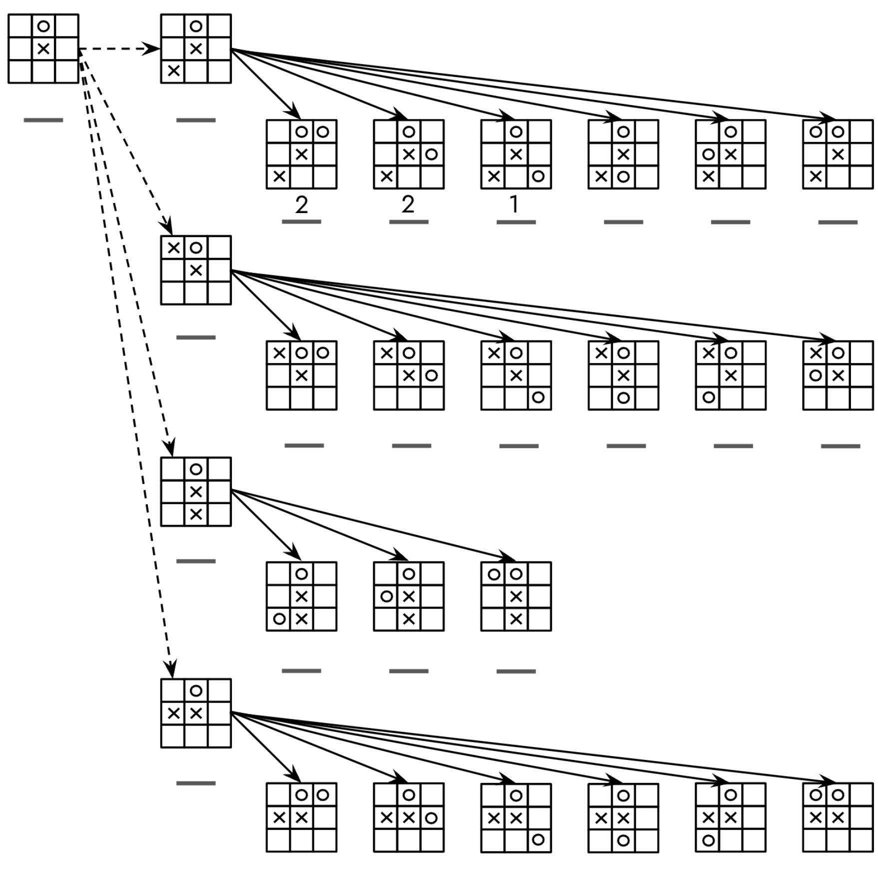



Figure 2: 2-ply deep search space to decide next move

(b) While solving (a), did you notice yourself doing any redundant work? Explain how you could have saved time by eliminating some options early.

## B Minimax with Pruning

1. Consider the minimax search tree shown in Figure 3. In the figure, black nodes are controlled by the MAX player, and white nodes are controlled by the MIN player. Payments (terminal nodes) are squares; the number within denotes the amount that the MIN player pays to the MAX player. Naturally, MAX wants to maximize the amount they receive and MIN wants to minimize the amount they pay.

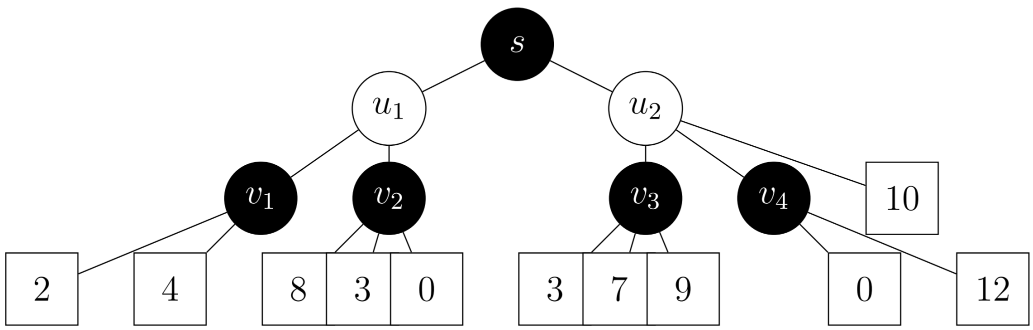



Figure 3: Minimax search tree

Suppose that we use the $\alpha$-$\beta$ Pruning algorithm reproduced in Figure 4.

```
def alpha_beta_search(state):
    v = max_value(state, -∞, ∞)
    return action in successors(state) with value v

def max_value(state, α, β):
    if is_terminal(state): return utility(state)
    v = -∞
    for next_state in expand(state):
        v = max(v, min_value(next_state, α, β))
        α = max(α, v)
        if v >= β: return v
    return v

def min_value(state , α, β):
    if is_terminal(state): return utility(state)
    v = ∞
    for next_state in expand(state):
        v = min(v, max_value(next_state, α, β))
        β = min(β, v)
        if v <= α: return v
    return v
```

Figure 4: $\alpha$-$\beta$ Pruning algorithm

(a) **Assume that we iterate over nodes from right to left**; mark with an 'X' all **arcs** that are pruned by $\alpha$-$\beta$ pruning, if any.

(b) Show that the pruning is different when we instead iterate over nodes from **left to right**. Your answer should clearly indicate all nodes that are pruned under the new traversal ordering.

## C Nonogram

1. Nonogram, also known as *Paint by Numbers*, is a logic puzzle in which cells in a grid must be colored or left blank according to numbers at the side of the grid. Usually, the puzzles are colored either in black or white, and the result reveals a hidden pixel-art like picture.

   Nonograms can come in different grid sizes. In this example, we will take a look at a $5 \times 5$ Nonogram. The game starts with an empty $5 \times 5$ grid. In the row and column headings, a series of numbers indicate the configuration of colored cells in that particular row and column. For example, on the fourth row, the "2" in the row header indicates that there must be 2 *consecutive* black cells in that row. In a more complicated case, such as in the second row, "1 2" indicates that there must be 1 black cell, followed by at least one blank cell, then 2 consecutive black cells. Similarly, in the first row, there must be 1 black cell, followed by at least one blank cell, another black cell, then at least one blank cell, and finally the last black cell. The rules apply similarly column-wise. For example, in the last column, there must be 4 consecutive black cells.

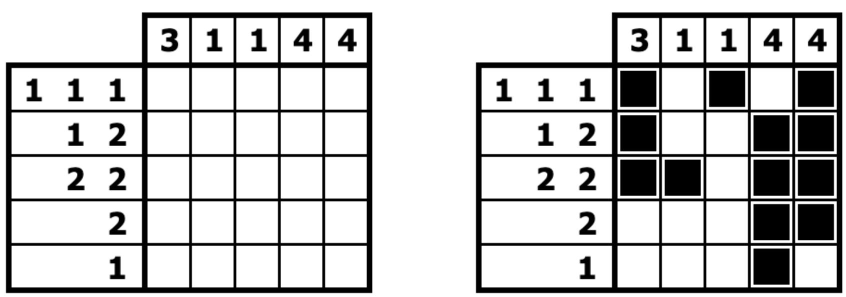



Figure 5: Nonogram

   To familarise yourself with the game, you may wish to play on this site.

   Given an empty $n \times n$ Nonogram with specified row and column color configurations, the challenging part of this game is to find out how you can color the cells in a way that satisfies **all** the constraints of the rows and columns.

   (a) Having learnt both systematic search and local search, you think that local search is more suitable for this problem. Give 2 possible reasons why systematic search might be a bad idea.

   (b) Based on the description of the problem above, propose a state representation for the problem.

   (c) What are the initial and goal states for the problem under your proposed representation?

# B

## a

(d) Define a reasonable transition function to generate new candidate solutions.

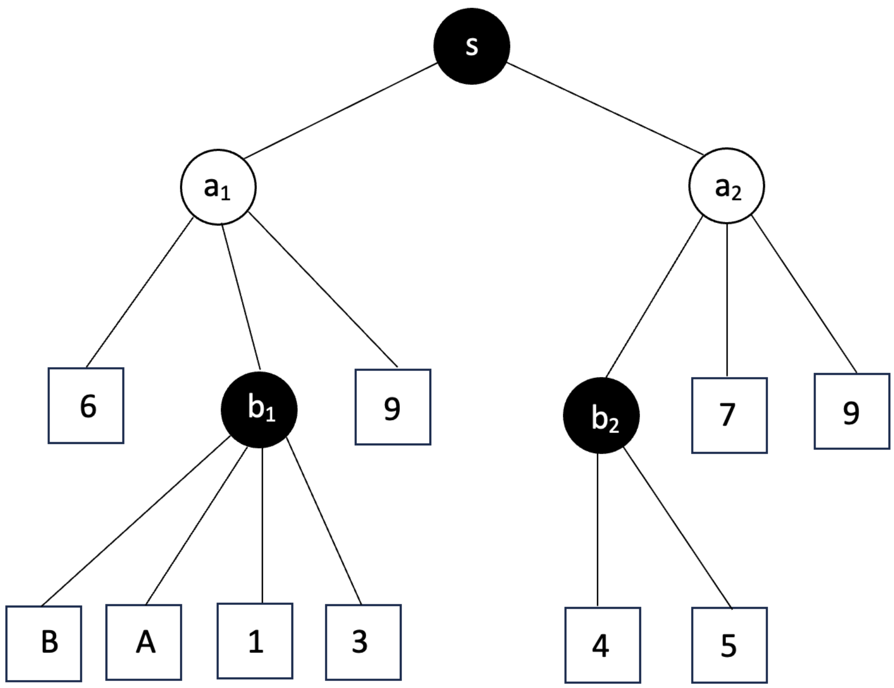
(e) Define a reasonable heuristic function to evaluate the "goodness" of a candidate solution. Explain how this heuristic can also be used as a goal test to determine that we have a solution to the problem.

(f) Local search is susceptible to local optima. Describe how you can modify your solution to combat this.

## D (Additional) $\alpha$-$\beta$ Pruning

1. In order for node B to NOT be pruned, what values can node A take on?

   **NOTE:** Assume that we iterate over nodes from right to left. Black node represents MAX player, white node represents MIN player.



## Complexity analysis of Minimax and alpha beta pruning optimization

Solve the recurrence relation with respect to the search depth $n$:

$$T(n)$$