National University of Singapore
School of Computing
CS2109S: Introduction to AI and Machine Learning
Semester I, 2023/2024

Tutorial 3
Adversarial Search and Local Search

These questions will be discussed during the tutorial session. Please be prepared to answer them.

Summary of Key Concepts

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

1. Adversarial Search
   (a) Minimax Algorithm
   (b) α-β Pruning
2. Local Search

Problems

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 shows a reduced state space for the first three levels with the player making the first move using "x" and the opponent making the next move with "o". Assume that the following heuristic evaluation function is used at each leaf node n:

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

   where $P(n)$ is the number of possible winning lines for the player while $O(n)$ is the number of possible winning lines for the opponent. A possible winning line for the player is a line (horizontal, vertical or diagonal) that either contains nothing or "x". For the opponent, it is either nothing or "o" in the line. Thus, for the leftmost leaf node in Figure 2, $Eval(n) = 6 - 5 = 1$.

   (a) Use the minimax algorithm to determine the first move of the player, searching 2-ply deep search space shown in Figure 2.

   (b) 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 of the filled leaf nodes and determine the second move of the "x" player in Figure 3 (searching 2-ply deep).

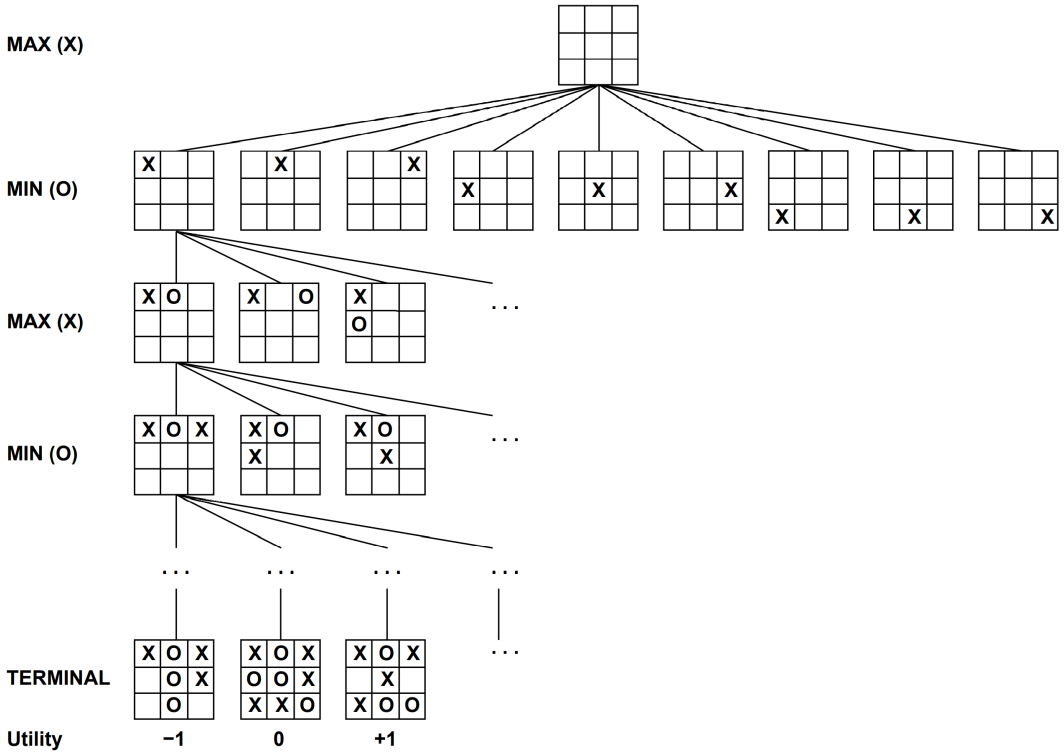


Figure 1: Search space for Tic-Tac-Toe.

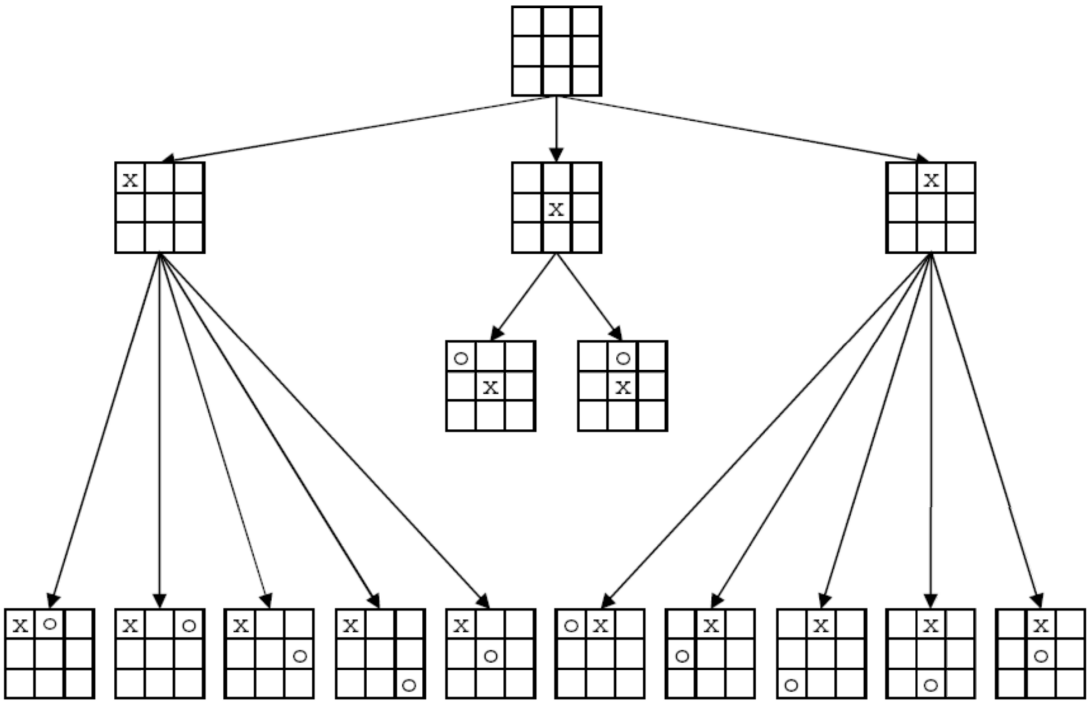


Figure 2: First move 2-ply deep search space solution

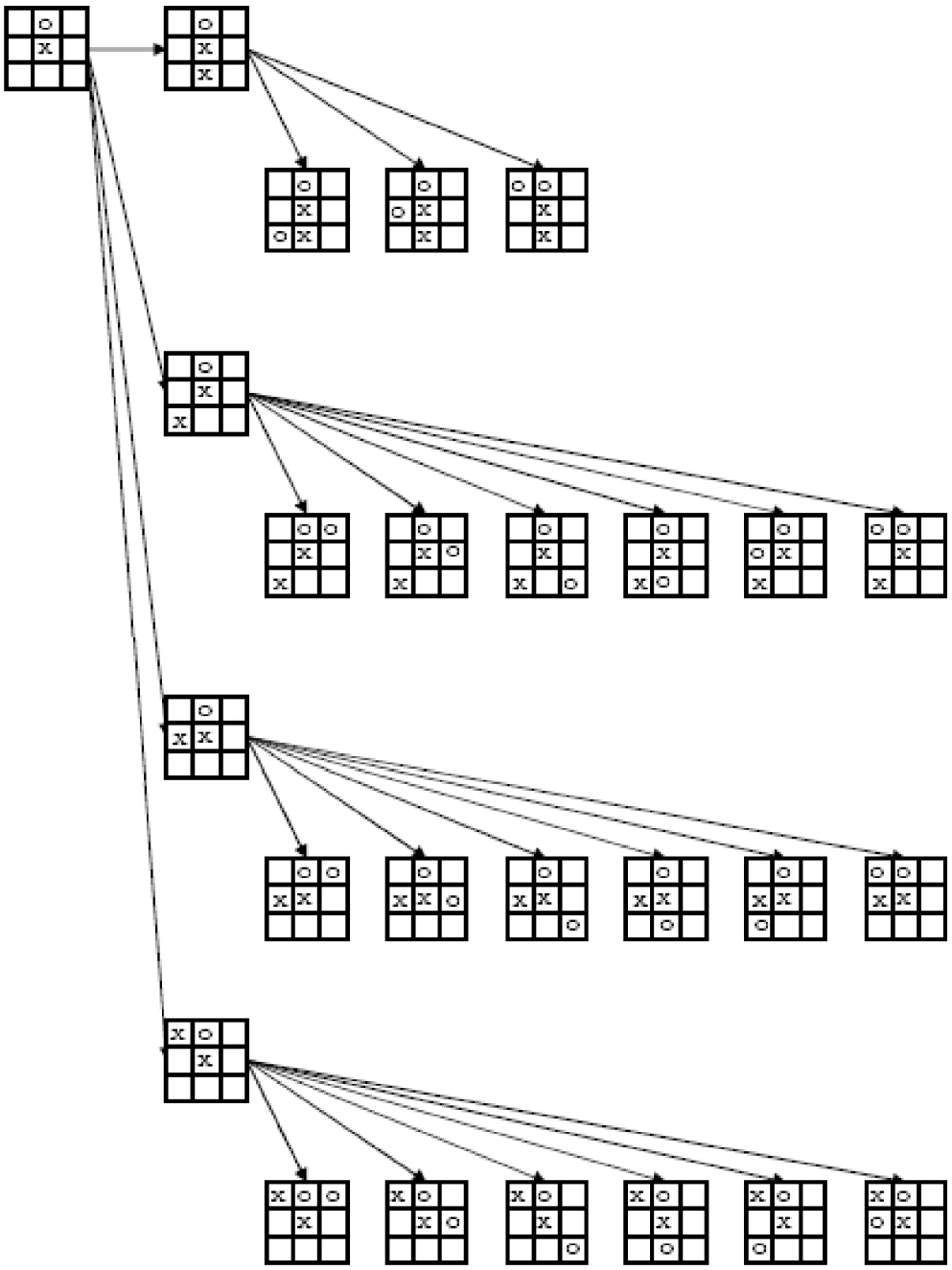


Figure 3: Second move 2-ply deep search space solution

2. Consider the minimax search tree shown in Figure 4. 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 (an amount of 0 means that MIN pays nothing to MAX). Naturally, MAX wants to maximize the amount they receive and MIN wants to minimize the amount they pay.

   Consider the minimax tree given in Figure 4.

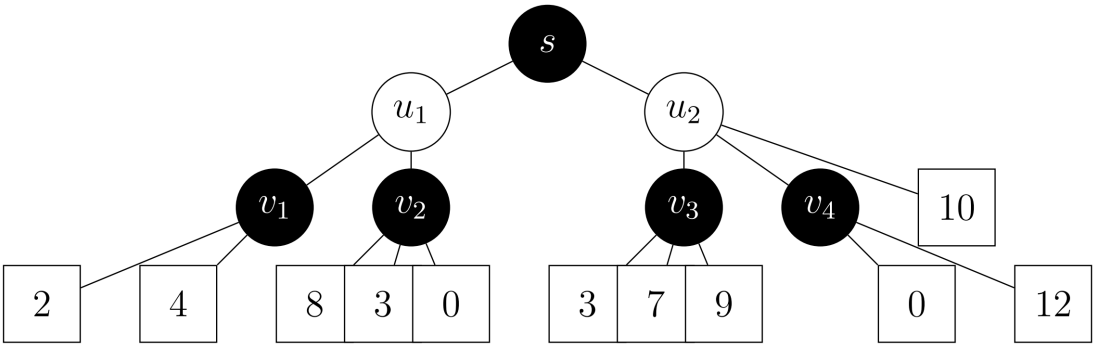


Figure 4: Minimax search tree

Suppose that we use the α-β Pruning algorithm reproduced in Figure 5.

```
function ALPHA-BETA-SEARCH(game, state) returns an action
  player ← game.TO-MOVE(state)
  value, move ← MAX-VALUE(game, state, −∞, +∞)
  return move

function MAX-VALUE(game, state, α, β) returns a (utility, move) pair
  if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
  v ← −∞
  for each a in game.ACTIONS(state) do
    v2, a2 ← MIN-VALUE(game, game.RESULT(state, a), α, β)
    if v2 > v then
      v, move ← v2, a
      α ← MAX(α, v)
    if v ≥ β then return v, move
  return v, move

function MIN-VALUE(game, state, α, β) returns a (utility, move) pair
  if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
  v ← +∞
  for each a in game.ACTIONS(state) do
    v2, a2 ← MAX-VALUE(game, game.RESULT(state, a), α, β)
    if v2 < v then
      v, move ← v2, a
      β ← MIN(β, v)
    if v ≤ α then return v, move
  return v, move
```

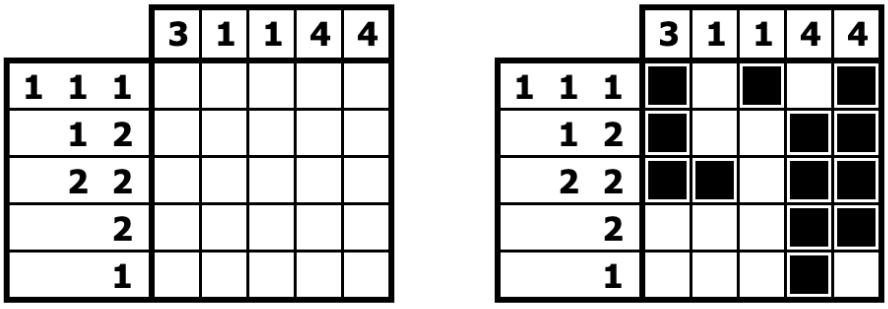
Figure 5: α-β Pruning algorithm

(a) **Assume that we iterate over nodes from right to left**; mark with an 'X' all **arcs** that are pruned by α-β 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.

3. 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 5×5 Nonogram. The game starts with an empty 5×5 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.



(a) Initial Nonogram (b) Solved Nonogram

Figure 6: 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 informed search and local search, you think that local search is more suitable for this problem. Give 2 possible reasons why informed search might be a bad idea.

   You decide to formulate the solution as a local search problem, which involves 3 steps:

   i. Define a candidate solution

   ii. Define a transition function to generate new candidate solutions

iii. Define a heuristic to evaluate the "goodness" of a candidate solution.

All 3 need to be designed in tandem because they have an impact on each other.

(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?

(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 minimas. Describe how you can modify your solution to combat this.

4. (Additional) 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.