CS2109s Tutorial 3

by Lee Zong Xun

Feedback for PS2

- Explaining the consistency and admissibility of your proposed heuristic.
  - Some of you provided really well-thought answers, while some of you wrote down specific examples.
  - Generally, we try to prove consistency by induction. Why? Induction prove for all n, the heuristic is consistent and this is a stronger proof than proving for a single n.
- Explaning heuristic for Local Search
  - Don't have to explain admissibility and consistency.
  - We don't even know how the goal state looks like! How would we know if the estimation we provide is optimal?

Recap

- Decision Tree is a representation of a function that maps a vector attributes to a single value.

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

... (cont.)

Generating a decision tree, t_2 from a training set formed by another decision tree, t_1 will generate a tree that is logically equivalent, not equal! Two attributes might have the same IG, so its 50/50 who gets picked as root.

Recap

- Information theory

Choose a root that offers the most information gain. Entropy is defined to be the measure of \textbf{impurity}. Entropy can be found using the following:

$$I(P(v_1), \ldots, P(v_n)) = -\sum_{i=1}^{n} P(v_i) log_2 P(v_i)$$

A chosen attribute, A, splits the training set E into subsets $E_1, \ldots, E_v$, where A has v distinct values.

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i})$$

... (cont.)

Information gain is given by

$$IG(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - remainder(A)$$

The main intuition is that even partitions offer more information gain than uneven partitions.

Recap

- Performance measures
  - Confusion matrix is a useful visualization to measure the accuracy, precision and recall of the model.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

  - Using precision and recall, we can compute the **F1 score**.

$$F1 = \frac{2TP}{2TP + FN + FP}$$

  - Statistically speaking, the measure is more robust.

Recap

Minimax algorithm

Depth first exploration. Includes the state of the game, and the player's turn.

$$Minimax(s) = \begin{cases} Utility(s, max), & \text{if } is-terminal(s) \\ max_{a \in A(s)} Minimax(R(s,a)), & \text{if } tm(s) = max \\ min_{a \in A(s)} Minimax(R(s,a)), & \text{if } tm(s) = min \end{cases}$$

Optional

Does a translation or scale of the values at the leaves affect the final outcome ? 🤔

Question 1 (a)

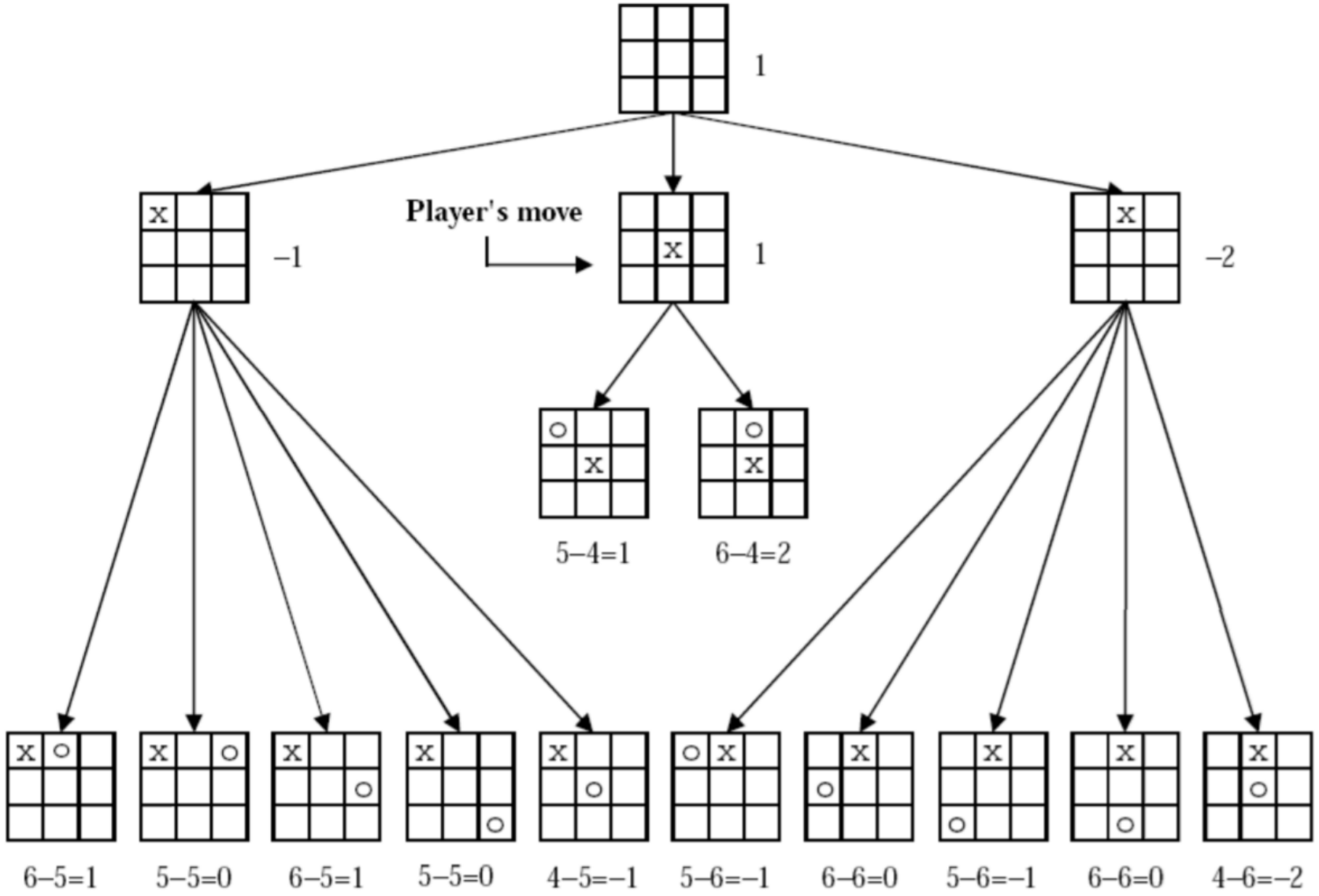
Assume that the following heuristic is used at each leaf node n:

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

where $P(n)$ is the number of possible winning lines for the player and $O(n)$ is the number of possible winning lines for the opponent.

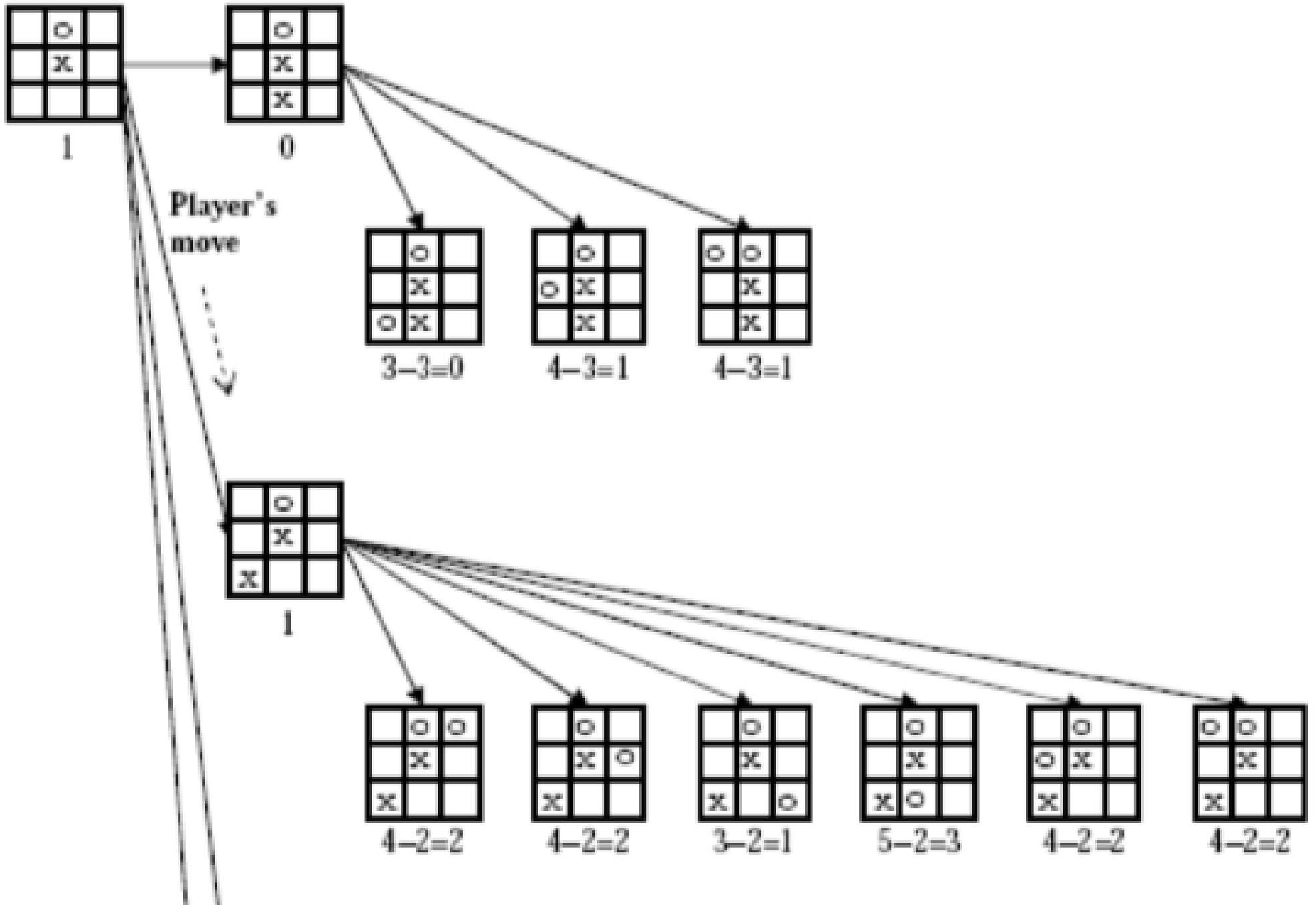
Use the minimax algorithm to determine the first move of the player, searching 2-ply deep search space shown in Figure 1.

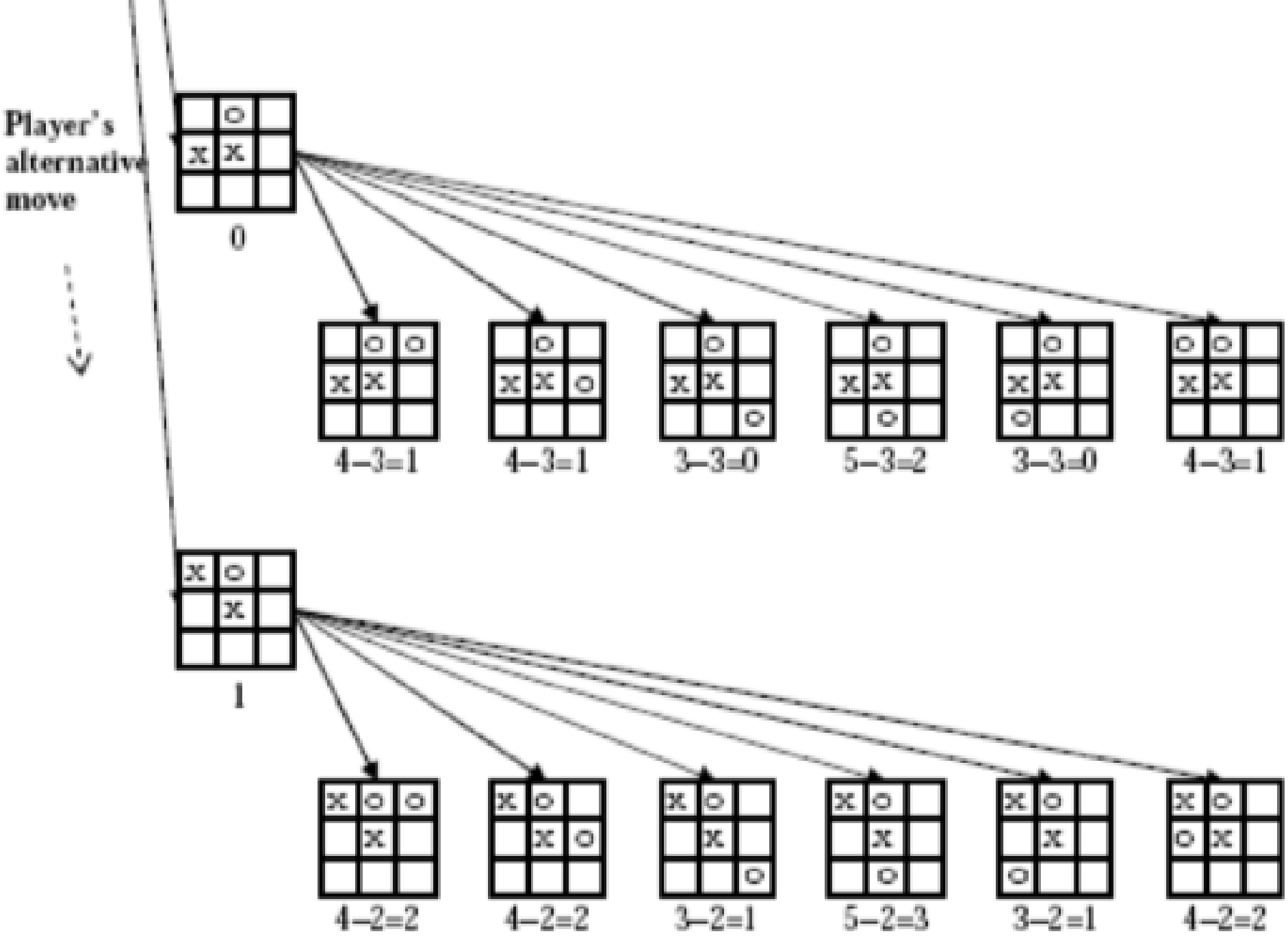
Solution


1
Player's move
–1
1
–2
5–4=1
6–4=2
6–5=1
5–5=0
6–5=1
5–5=0
4–5=–1
5–6=–1
6–6=0
5–6=–1
6–6=0
4–6=–2

Question 1 (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 (searching 2-ply deep).

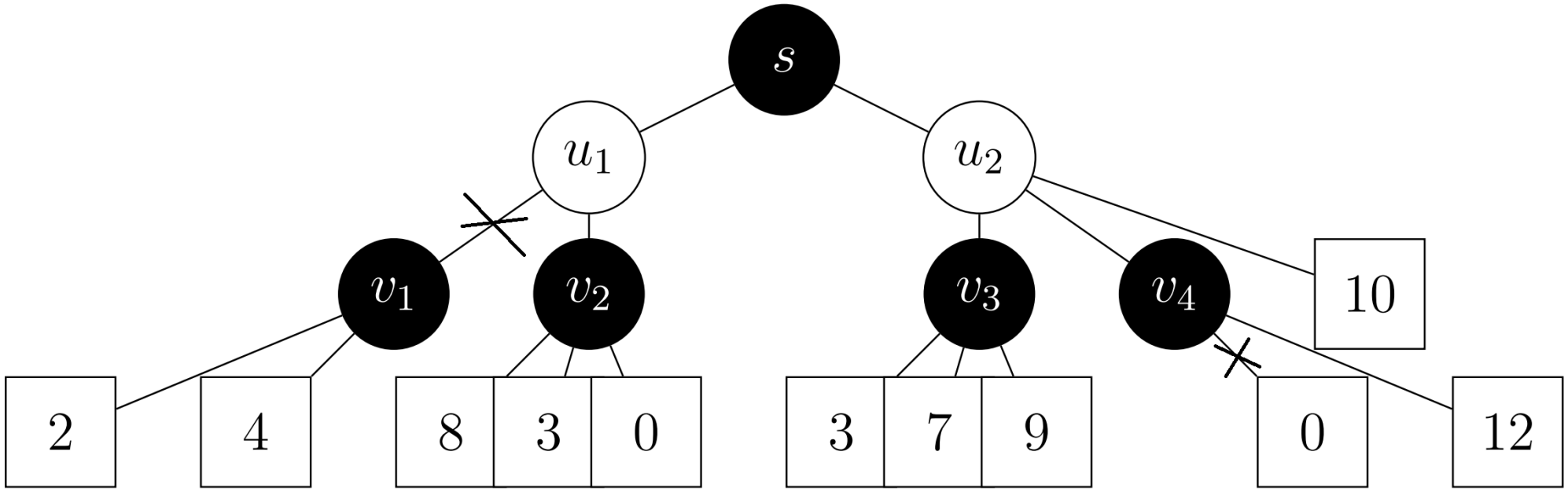

1
0
Player's
move
3–3=0
4–3=1
4–3=1
1
4–2=2
4–2=2
3–2=1
5–2=3
4–2=2
4–2=2


Player's alternative move
0
4–3=1
4–3=1
3–3=0
5–3=2
3–3=0
4–3=1
1
4–2=2
4–2=2
3–2=1
5–2=3
3–2=1
4–2=2

Question 2

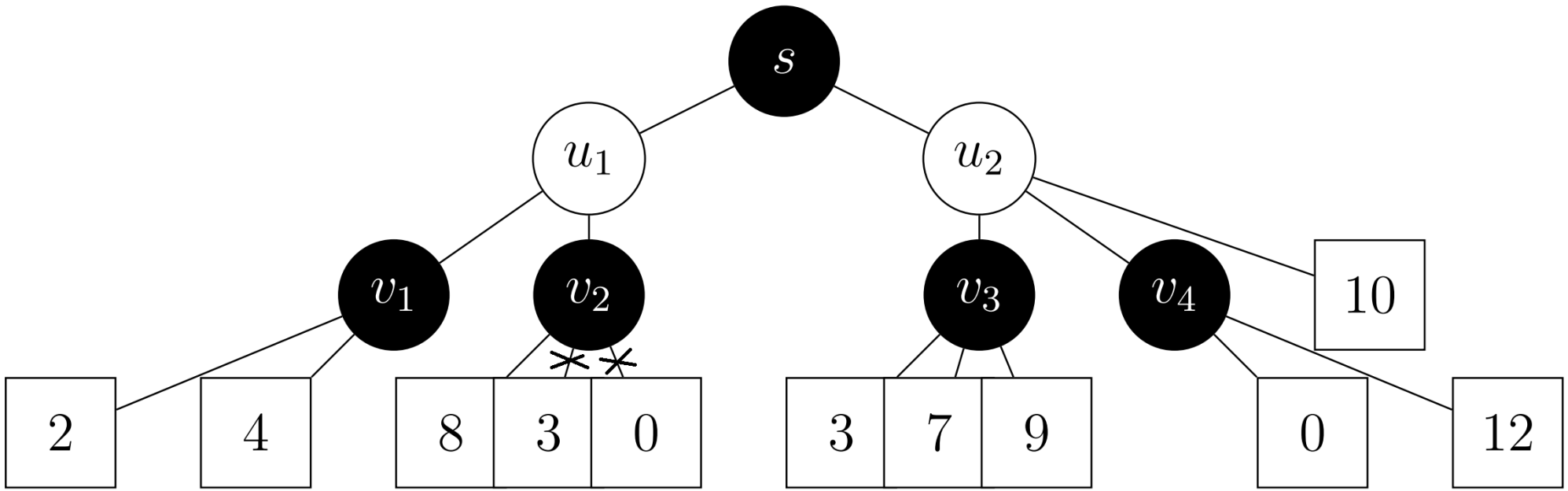
Assume that we iterate over nodes from right to left; mark with an 'X' all **arcs** that are pruned by α-β pruning, if any.

Question 2 (cont.)



Question 2(b)

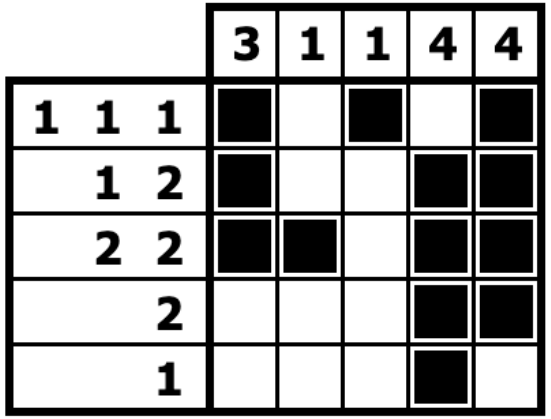
Show that the pruning is different when we instead iterate over nodes from left to right. Your answer should clearly indicate all nodes that are pruned under the new traversal ordering.


s
u_1
u_2
v_1
v_2
v_3
v_4
10
2
4
8
3
0
3
7
9
0
12

Question 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.

Visuaizer: https://handsomeone.github.io/Nonogram/#game



Question 3

Given an empty n×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.

- 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.**

Question 3 (...cont.)

Some questions to ask yourself:

1. how huge is the search space? HUGE $O(2^{n^2})$
2. does the path to solution matters? No!
3. is there even a solution? 🤔

Question 3(b)

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

> Use an $n \times n$ boolean matrix, where each element is either true (if the corresponding cell is colored) or false (if the corresponding cell is not colored).

Question 3(c)

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

The initial state is an $n \times n$ boolean matrix with m entries set to true, while the rest of the entries are set to false.

But how do we choose m?

To choose the positions of the m entries that are set to true, we can do so randomly, while ensuring that the entries adhere to the row constraints.

What is the goal state then?

Question 3(d)

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.

The min-conflict heuristic. If there are no violations, then we have a valid solution and the problem is solved.

Notice that we do not need to count the number of instances where the constraints on the row configurations are violated, because by design, we have already ensured the adherance of the entries to the row constraints.

Question 3(e)

Define a reasonable transition function to generate new candidate solutions.

Given the current candidate, we can pick a random row and generate the list of neighbours with the corresponding row permuted. Then, we can move to the neighbour with the lowest cost.

> Why not generate all possible permutations and pick the one with the lowest cost?

3(f)

Local search is susceptible to local minimas. Describe how you can modify your solution to combat this.

- introduce random restarts by repeating local search from a random initial state.
- use simulated annealing search to accept a possibly ``bad" state with a probability that decays over time
- beam search to perform k hill-climbing searches in parallel.