Search Questions

Alexander Resare, Nikita Suprun

March 27, 2024

Q1

An arbitrary state is generally defined as an observation of the hook **hook_positions** and fishes **fish_positions**. The time of the game also makes up for a state variable. Assuming player **max** starts and the player takes turns making one move each for the duration of the game, the player turn **player** can be derived any point in time, just by looking at the time. Therefore, at its most fundamental level, each state can be regarded as a combination of the following attributes:

- **hook_positions**
- **fish_positions**
- **time**
- **player scores**

The remaining state attributes do not define the state since they can be derived from the other variables.

The initial state is given by some data preloaded into the fishing_derby game. Most importantly for the initial state, the time is 0 and the time remaining is at 100% The player whose turn it is, is player **max**.

The transition function is a mapping between all possible moves and the state at the next time. Given that the possible moves are at least 3 and at most 5, the transition function results in a set of three to five new states, given the current state.

Q2

For the terminal state is achieved when, either the set of fish positions is empty, since all the fish have been caught, or the time remaining is 0. Two outcomes are possible, MIN wins or MAX wins.

Q3

Given a state s with player turn A and a successor state s' with player turn B, assume A wants to maximize the heuristic function v, while B seeks to minimize v. Now let $v(A,s) = \text{Score}(A,s) - \text{Score}(B,s)$. Assume for the successor state s' that $\text{Score}(A,s') > \text{Score}(A,s)$ while $\text{Score}(B,s') = \text{Score}(B,s)$. This means that:

$$\begin{aligned} v(B,s') - v(A,s) = &- \text{Score}(A,s') - \text{Score}(B,s') - \big(\text{Score}(A,s) - \text{Score}(B,s)\big) \\ =&\text{Score}(A,s') - \text{Score}(A,s) > 0 \\ &\implies v(B,s') > v(A,s) \end{aligned}$$

A has thereby managed to achieve a state that increased it's own score as well as the utility function, adhering to the heuristic function's goal. For B it can be shown that increasing B's score in the successor state, reduces the value of the heuristic function, also adhering to the utility function criteria. This is why the example heuristic function is a good utility function.

Q4

Let us first understand what a utility function is. A utility function determines the final numeric value to a player when the game ends in terminal state s.

When there is no fish left or the time has run out, the player scores are set. Hence, v gives the best approximation of the utility function when the all the fish has been caught. In fact, $v=$ utility function for the terminal states.

Hence, v gives the best approximation of the utility function γ if $v \approx \gamma$ when the terminal state has been reached.

Q5

Assume the utility function v is given as the one in **Q3** and the player scores are $A : 23$ and $B : 21$. Also assume for this example that the last remaining fish in the game is a fish of score 4 that has been caught by B and is one position away from B. In the next state, there are no remaining fish available so the terminal state has been reached. For the terminal state, we now have the scores $A : 23$ and $B : 25$, meaning $v(B, s_t') < 0$ and B has won.

Q6

In chess such a heuristic is problematic, since even if A has 10 winning states, and only 1 loosing state, say B can check mate A in 1 move, then it is the move B will most likely take and A will loose despise the heuristic having a large positive value. As mentioned in the instructions, the heuristic tells nothing about the distribution of the states and assumes B is using brute-force which would indicate a uniform distribution of terminal states.

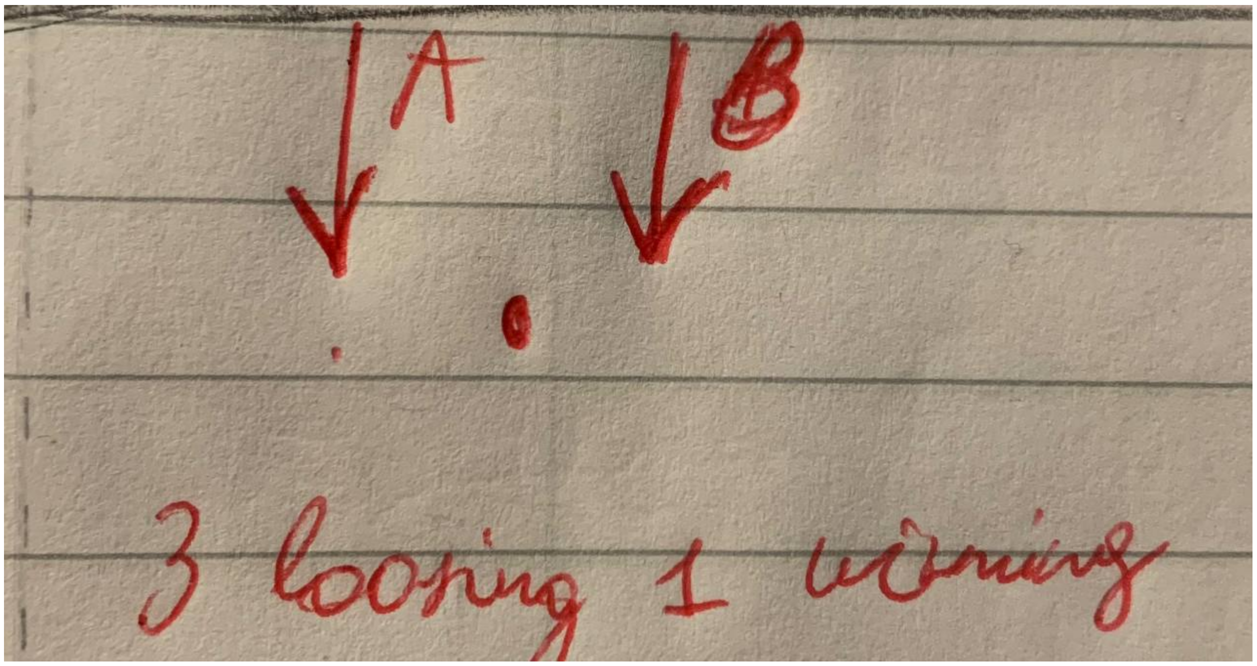

Figure 1: heuristic

For the example of the fishing derby, assume $A\&B$ are located in the middle of the map. To the left of A and farther left of B, the last fish is located. If A adheres to the η heuristic function, A will try and keep B from reaching the fish, spending most of it's moves keeping B to the right side of the map and thereby limiting it's possible ways to catch the last fish. Meanwhile, B might instead focus on adjusting it's hook in accordance with the fish's y-position. Eventually, B might accidentally catch the fish and win because A was too focused on limiting the amount of ways B could win.

The situation from **Q5** might happen with η, as say A has 3 points more than be and there is one fish left that costs 5 points, shown in Figure 1, assume further there are 2 turns left and it is now A's turn. For the η heuristic function, it is arbitrary whether A chooses to go right$\rightarrow$down or down$\rightarrow$right. However, it is obvious that if B goes right and the fish stays put, there are 0 ways A can win, while B is one move away from winning the game. The point to clarify here is that η fails to predict this outcome, since it only focuses on the number of winning and loosing states at present, but not their true distribution.

$\alpha - \beta$ pruning

The effectiveness of alpha–beta pruning is highly dependent on the order in which the states are examined. For example, in Figure 6.5(e) and (f), we could not prune any successors of D at all because the worst successors (from the point of view of MIN) were generated first.

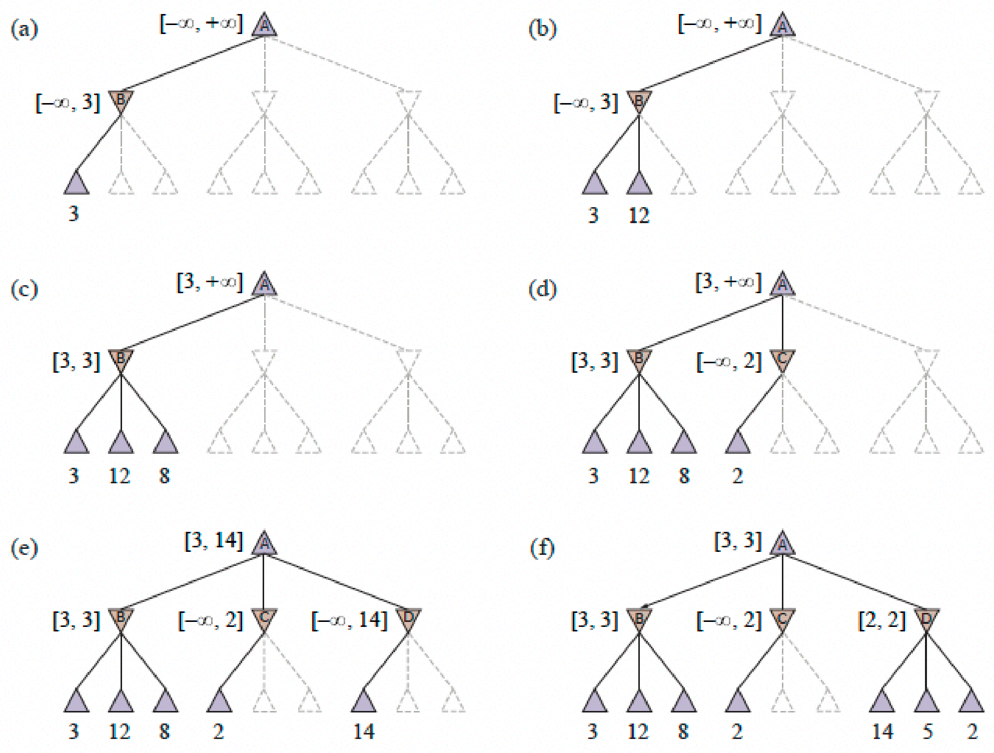

Figure 6.5 Stages in the calculation of the optimal decision for the game tree in Figure 6.2. At each point, we show the range of possible values for each node. (a) The first leaf below *B* has the value 3. Hence, *B*, which is a MIN node, has a value of *at most* 3. (b) The second leaf below *B* has a value of 12; MIN would avoid this move, so the value of *B* is still at most 3. (c) The third leaf below *B* has a value of 8; we have seen all *B*'s successor states, so the value of *B* is exactly 3. Now we can infer that the value of the root is *at least* 3, because MAX has a choice worth 3 at the root. (d) The first leaf below *C* has the value 2. Hence, *C*, which is a MIN node, has a value of *at most* 2. But we know that *B* is worth 3, so MAX would never choose *C*. Therefore, there is no point in looking at the other successor states of *C*. This is an example of alpha–beta pruning. (e) The first leaf below *D* has the value 14, so *D* is worth *at most* 14. This is still higher than MAX's best alternative (i.e., 3), so we need to keep exploring *D*'s successor states. Notice also that we now have bounds on all of the successors of the root, so the root's value is also at most 14. (f) The second successor of *D* is worth 5, so again we need to keep exploring. The third successor is worth 2, so now *D* is worth exactly 2. MAX's decision at the root is to move to *B*, giving a value of 3.

Figure 2: a, b pruning algorithm