Competitive heuristic search

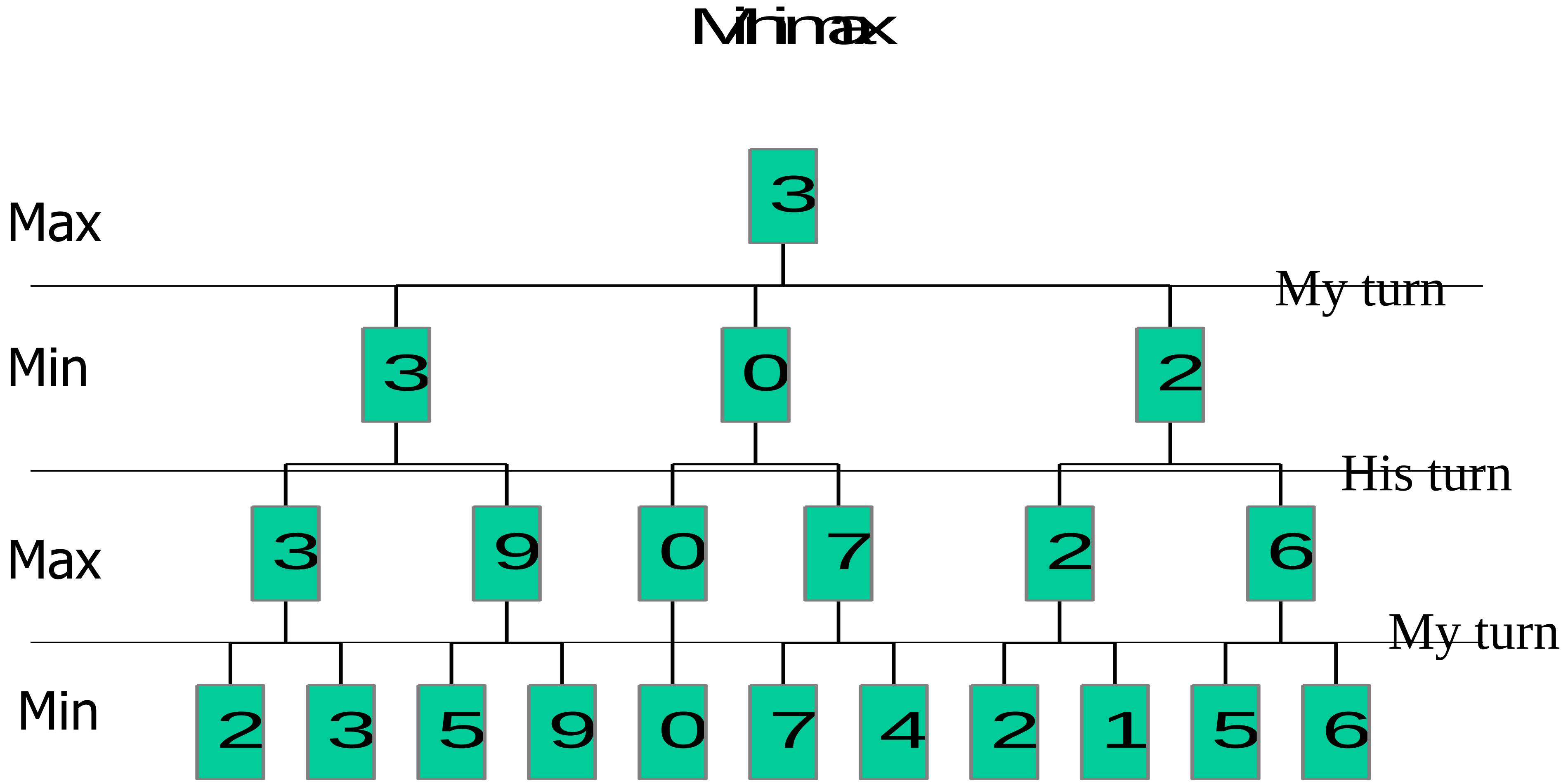
Minimax (exhaustive)

- Assumes the opponent is using the same knowledge of the space as you are.
- Each layer in the SS is labeled MAX (us) or MIN (the opponent).
- Each leaf node is given 1 (win) or 0 (loss).
- If parent state is at a MAX level, give it the maximum value from its children. If MIN, give it the minimum value.
- Follow the 1s to win!

Minimax to fixed ply depth

- Good for games in which exhaustive search is not possible
- Minimax to chosen depth, then apply evaluation heuristic.
- Propogate values up using normal minimax.

Minimax to fixed depth



Problem: Horizon effect

- If search is not exhaustive, it will necessarily have a maximum depth, and have to evaluate nodes at that depth with some (imperfect) evaluation function.
- It’s possible that some node beyond this horizon (perhaps even the very next successor!) would win or lose the game – but we can’t see it!

Alpha-beta pruning

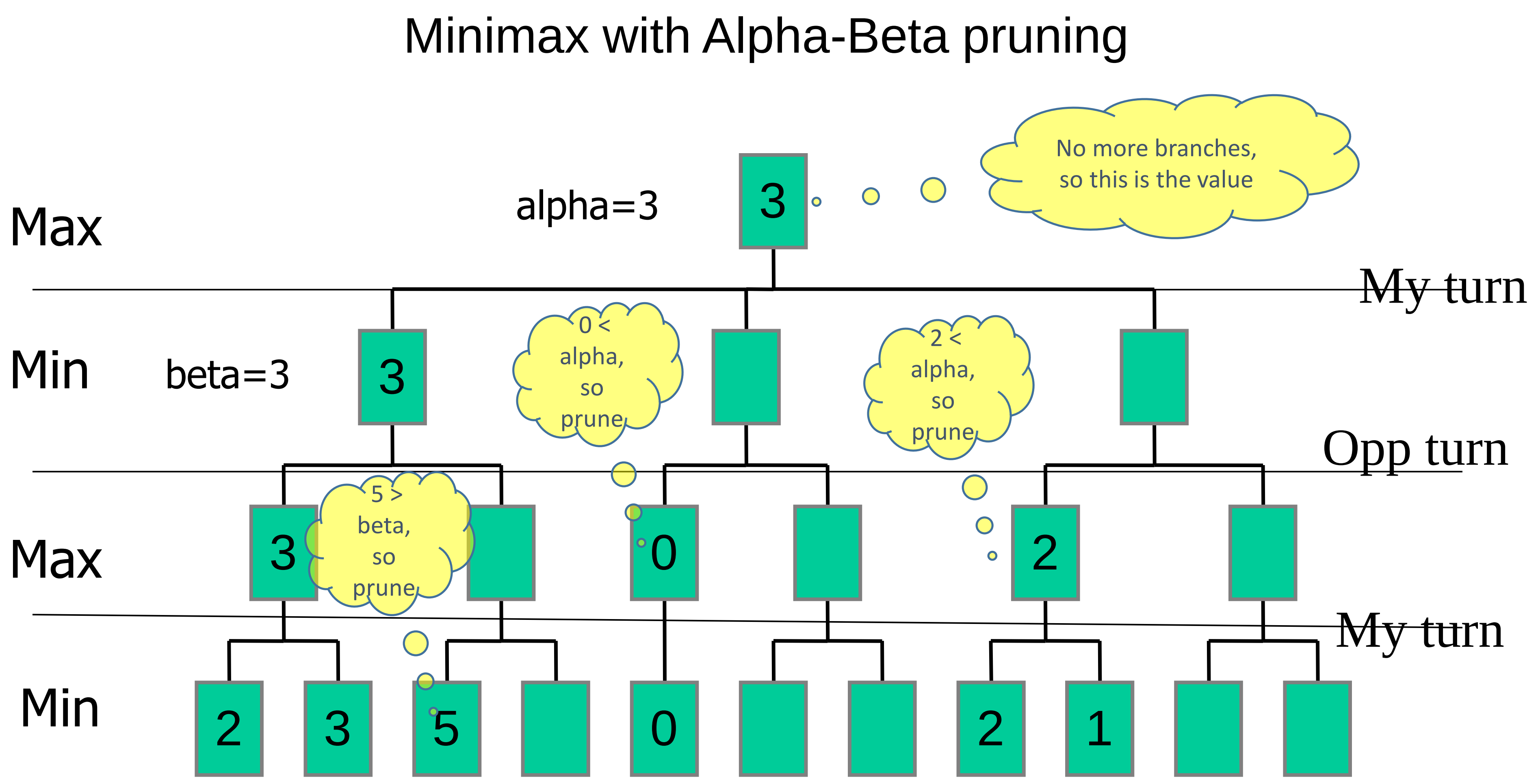
- A way of pruning the search-space on Minimax. Roughly halves the space. Alpha and beta are *bounds* for the minimax values.
- Alpha: at a MAX node, the largest minimax value so far, amongst the daughters. That node will not end up having a value *less than* alpha.
- Beta: at a MIN node, the smallest minimax value so far, amongst the daughters. That node will not end up having a value *greater than* beta.

How to prune

Do DFS, to maximum depth. Keep track of alpha and beta, tentative values for MAX and MIN nodes, respectively. For each node with a known minimax value:

- If a MIN node, and value is greater than the beta above, no need to search parent (MAX) node and below.
- If a MAX node, and value is less than the alpha above, no need to search parent (MIN) node and below.

With alpha-beta pruning (animated)



Advantages of alpha-beta pruning

- Can significantly reduce the search space (although worst case is still approx O(b^d).) [Minimax is a variation on depth first]
- Search space reduction depends crucially on ordering of offspring states.
- For chess, a simple ordering heuristic (captures, threats, forward, backward) gets close to O($b^{d/2}$), so can look almost twice as far ahead!

Exercise (optional)

- Use minimax/alpha-beta for tic-tac-toe
- What is the average branching factor (approx)?
- How many moves deep is the the shortest game? The longest?
- Use heuristic(s) to evaluate the first player’s (I.e. your) third move. Use minimax + alpha-beta pruning to percolate the values, and choose the right move.

Samuel’s checkers program (1952, beat a champ in 1962)

- Used a wide variety of heuristics (control of center, # of pieces, etc), summed in a polynomial, with fixed-ply-depth minimax.
- System adjusted the coefficients on the polynomial if unsuccessful (reducing variance) or successful (increasing variance). **Problem**: which heuristic was at fault?
- Played very well, but no global strategy, so vulnerable to strategies that “understood” its approach
- Vulnerable to the *horizon effect*.

Chinook

- Checkers program, uses alpha-beta search. Became world checkers champion.
- Large database (44 billion) of endgame positions (8 or fewer pieces).
- Can announce a win as early as move 5!
- Checkers was exhaustively solved in 2007 by Chinook's creator, Schaeffer.

Deep Blue

- In 1997, beat world champion Kasparov
- Can search up to 330 million nodes per second
- Search can reach up to 40 plies
- Has 8000 feature evaluation function
- Huge endgame database
- Pruning *very* important – reduces effective branching factor from ~35 to <3