IMPERFECT REAL-TIME DECISIONS

- Moves must be made in a reasonable amount of time.
- usually it is not feasible to consider the whole game tree (even with alpha-beta)
- Programs should cut the search off at some point earlier and apply a heuristic evaluation function to states in the search, effectively turning non terminal nodes into terminal leaves.

Alter min-max or alpha-beta in 2 ways:

- Replace the utility function by a heuristic evaluation function EVAL, which estimates the position's utility.
- Replace the terminal test by a cutoff test that decides when to apply EVAL.

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

Replace using **heuristic evaluation function EVAL**

$$\text{H-MINIMAX}(s,d) = \begin{cases} \text{EVAL}(s) & \text{if CUTOFF-TEST}(s,d) \\ \max_{a \in Actions(s)} \text{H-MINIMAX}(\text{RESULT}(s,a), d+1) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{H-MINIMAX}(\text{RESULT}(s,a), d+1) & \text{if PLAYER}(s) = \text{MIN.} \end{cases}$$

EVALUATION FUNCTION:

- An evaluation function returns an estimate of the expected utility of the game from a given position.

How do we design good evaluation functions?

1) The evaluation function should order the terminal states in the same way as the true utility function.

2) The computation must not take too long.

3) For nonterminal states, the evaluation function should be strongly correlated with the actual chances of winning.

Two ways to design an Evaluation Function:

i. Expected value

ii. Weighted linear function

i. Expected Value:

The evaluation function cannot know which states are which, but it can return a single value that reflects the proportion of states with each outcome.

Where,

- EV – the expected value
- P(X) – the probability of the event
- n – the number of the repetitions of the event

Example: How many heads would you expect if you flipped a coin twice?

Solution: X = number of heads = {0, 1, 2}

p (0)=1/4,

p (1)=1/2, p (2)=1/4

Weighted average = 0*1/4 + 1*1/2 + 2*1/4 = 1

ii. **Weighted linear function:** We can compute separate numerical contributions from each feature and then combine them to find the total value.

- EV is the probability-weighted average of all possible events.
- Therefore, the general formula to find the EV for multiple events is:

Where,

- EVAL(s) – the expected value
- wi is a weight and each
- fi is a feature of the position

Example: Three coin tosses, the expected value for the number of heads is 3 x (1/2) = 1.5.

Outcome	# Heads	Probability	# Heads x Probability
TTT	0	1/8	0
TTH	1	1/8	1/8
THT	1	1/8	1/8
THH	2	1/8	2/8
HTT	1	1/8	1/8
HTH	2	1/8	2/8
HHT	2	1/8	2/8
HHH	3	1/8	3/8
			Total: 1.5

Cutting off search:

To modify ALPHA-BETA-SEARCH:

1) Replace the two lines that mention TERMINAL-TEST with

2) Arrange for some bookkeeping so that the current depth is incremented on each recursive call.

- The most straightforward approach: set a fixed depth limit so that CUTOFF-TEST (state, depth) returns true for all depth greater than some fixed depth d.
- A more robust approach: apply iterative deepening.
- A forward-pruning version of alpha-beta search that uses statistic gained from prior experience to lessen the chance that the best move will be pruned.

Search versus lookup:

- Many game programs pre-compute tables of best moves in the opening and endgame so that they can look up a move rather than search.

Eg: BFS, DFS

- For the opening (and early moves), the program use table lookup, relying on the expertise of human and statistic from a database of past games.

Eg: Two player games, multiple player game.