

1 Game Playing, Cont'd

- Last time, we talked about how to build (learn) an evaluation function, and some of the gotchas for that.
- Another thing you have to specify is when to stop raw search and truncate tree – when to apply eval func.
- Simplest: time cutoff.
- Slightly smarter: iterative deepening w/ time cutoff.
- Problem: This still searches all of the tree uniformly and takes exponential time. Often, you want to focus search into high-productivity parts of the tree.
- Better: determine when further search is fruitless/low priority and when to stop descending.
- Problem: Hard to know when you should stop descending.
- E.g., suppose you're using weighted piece count as a board evaluator in chess, and you're up by a queen and decide to stop descending that branch b/c it looks like you're safely ahead.
- But maybe the queen will inevitably be captured on the next move. So the position is not *actually* as good as it appears.
- The game may change radically just *beyond* the depth of your search, so your evaluation at the truncation point could be very far off.
- In practice, this is often a big problem and leads game players into bad parts of the state space – false sense of security b/c they haven't looked far enough ahead.
- (This may or may not be a significant problem in Awari – I haven't thought through it enough to be sure. You should.)
- One way to avoid this is to avoid making your evaluation when the board is still unstable – e.g., when a capture is imminent or something.
- When the eval function settles down, it's called **quiescence** and the search that only makes an evaluation when the board is settled is *quiescence search*.
- Note that quiescence is a function of full board evaluation fn. Can be hard to know when board is quiescent. In practice, often restrict quiescence evaluation to a few, easy to evaluate factors (e.g., immediate captures).
- Another, very thorny problem, is **horizon effect**. This happens when some outcome can be *postponed* but is ultimately *inevitable*.
- E.g., pawn-to-queen promotion example from R&N.

- Problem is that search procedure can't necessarily recognize inevitability and so will tend to just push off outcome as long as possible. If pushes it beyond search depth, it basically willfully ignores possibility.
- This is a really nasty problem with no really good answer.
- One possibility is to follow **singular extensions** (a.k.a., **forcing paths**) – paths where one move is dramatically better than others.
- By following only single, clearly best move, can search much deeper (Deep Blue could go like 40+ ply along forcing paths, I think).

2 Games of Chance, Uncertainty, and Decision Theory

- So far, we've talked only about games with deterministic dynamics. E.g., in checkers, when you move a piece, you know exactly where it will end up, you know exactly what the board will look like, etc.
- What if you're in a game of chance? What if you *don't* know what will happen next? Then how should you act?
- Example: the chances of exactly hitting the powerball jackpot in the NM lottery are roughly 1 in 120 million (1 in 120,526,770, to be exact). If a ticket costs \$1, and the grand prize is \$10 million, should you play? What if the grand prize is \$50 million? \$200 million?
- Well... It depends.
- This falls into branch of study called **statistical decision theory**
- Have to know 4 things to make your decision:
 1. What **decisions** (choices, actions) can you make?
 2. What **outcomes** are possible? In this case, 1 win, 120 million-1 non-wins (assuming jackpot is the only prize).
 3. What is the **probability** of each outcome?
 4. What is the **utility** (a.k.a., value, prize, return, etc.) of each outcome?
- Rule is that we want to maximize the *average* return.
- (Big question about why that should be so. It's actually a subject of some argument. In one sense, if you repeat the trial an infinite number of times, the mean is the value you get back in the limit. Problem is that you never do anything an infinite number of times. Real answer is that it's tractable for us to analyze and, in real scenarios, it *does* lead to reasonable decisions in many cases.)
- So for the lottery question:

1. Decisions: play, not play.
2. Outcomes: Not play: single outcome (nothing happens). Play, 120 million+ outcomes (1 win, 120 million-1 lose)
3. Probability: not play: single outcome w/ Pr 1. PlayL each outcome has same prob (1/120 million)
4. Utility: not play: single outcome has utility 0. (You neither gain nor lose money.) Play: win has utility \$10 million-1, others have utility -\$1.

- So now the utility of our two actions are:

$$\begin{aligned}
 U(\text{notplay}) &= 1 \cdot 0 = 0 \\
 U(\text{play}) &= 9,999,999 \cdot \frac{1}{120,526,770} + -1 \cdot \frac{120,526,769}{120,526,770} = -0.917
 \end{aligned}$$

- I.e., *in the limit*, you will lose 91.7 cents if you play this lottery.
- Note that if the prize is *big enough*, the leading term will dominate the trailing term, and you'll make money.
- E.g., if the prize is \$ 1 *billion*, then you'll *win* about \$8.30 on average. (Lot of effort to go through, though.)
- Casinos exploit this property ruthlessly. If you look at how they set up the bets for their tables, *on average*, they win like 2 cents per transaction or something. So they lose plenty of *individual* transactions, but they rake in the dough *on average*.
- This is also how the lottery makes money (note that the lottery is a for-profit corporation. . .)
- Our old rule, for deterministic games was “choose the decision/action with the maximum value”
- Our *new* rule, for stochastic games is “choose the decision/action with maximum *expected* value”
- This is the principle of **maximum expected utility**.
- Chance introduces **chance nodes** into a minimax tree. Becomes an *expectiminimax* tree.
- Utility of a chance node is:

$$\begin{aligned}
 U(s) &= E_{s' \in \text{outcomes}}[U(s')] \\
 &= \sum_{s' \in \text{outcomes}} \text{Pr}[s'] U(s')
 \end{aligned}$$

- In particular, if you want to make a single decision, choosing $d \in D$ where D is the set of all possible decisions, then the max expected utility decision is:

$$\begin{aligned} d &= \arg \max_D E_{s' \in \text{outcomes}} [U(s') | d] \\ &= \arg \max_D \sum_{s' \in \text{outcomes}} \Pr[s' | d] U(s') \end{aligned}$$