Game playing II - Expectimax

Chapter 5

Adapted from slides kindly shared by Stuart Russell

Announcements

P1 (search) due Thu 10-04 at 5pm

Autograder will be run manually tonight and a few times tomorrow.

Project P2 Multi-Agent Pac-Man coming later this week.

◇ Builds on Search in Pac-Man

◇ Now with Ghosts! Minimax, Expectimax, Evaluation

Office hours for me on Thursday 2:30-3:30 and Friday 1:30-2:30, ECST 121

Thanks to Dan Klein for some of these slides

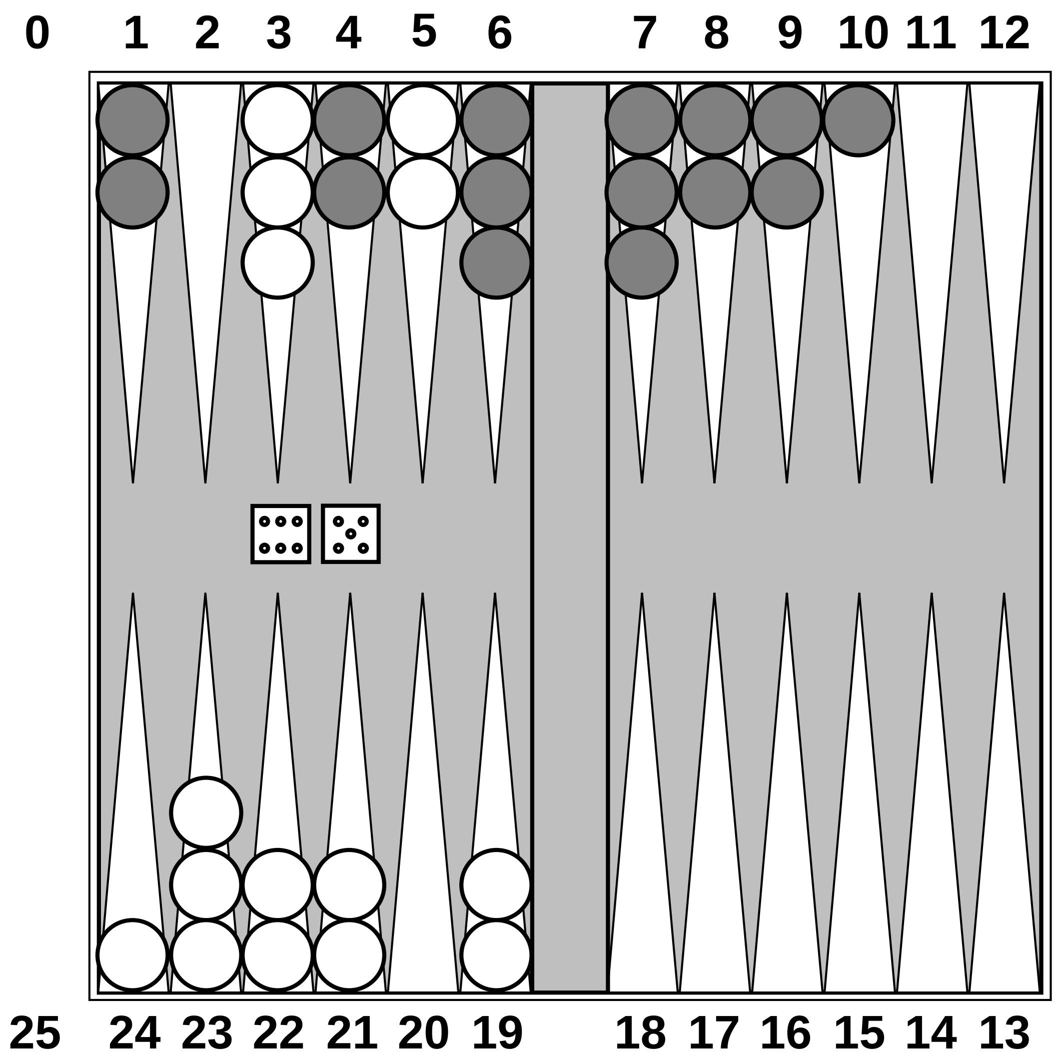
Hand quizzes back

Outline

◊ Games of chance

◊ Games of imperfect information

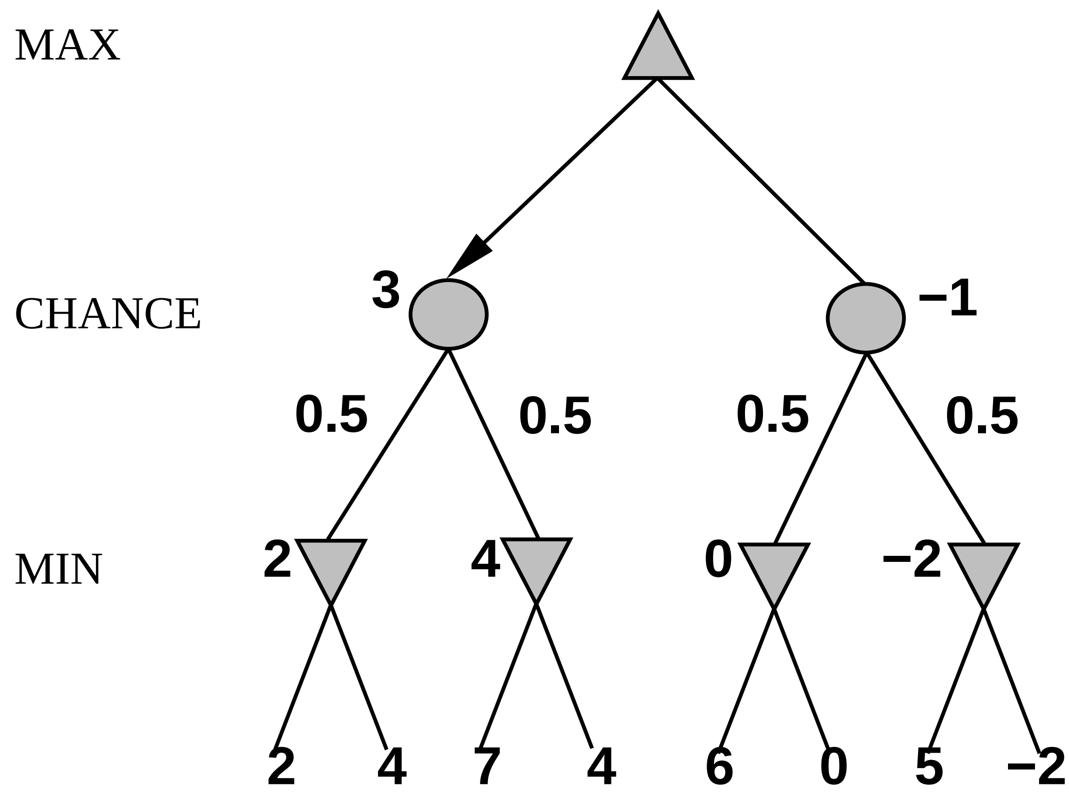
Nondeterministic games: backgammon



Nondeterministic games in general

In nondeterministic games, chance introduced by dice, card-shuffling

Simplified example with coin-flipping:



Expectimax Search

For “chance” nodes: Calculate expected utilities

I.e. take weighted average (expectation) of values of children

Algorithm for nondeterministic games

EXPECTIMAX (or EXPECTIMINIMAX etc) gives perfect play

Just like MINIMAX, except we must also handle chance nodes:

```
...
if state is a MAX node then
    return the highest EXPECTIMINIMAX-VALUE of SUCCESSORS(state)
if state is a MIN node then
    return the lowest EXPECTIMINIMAX-VALUE of SUCCESSORS(state)
if state is a chance node then
    return weightavg of EXPECTIMINIMAX-VALUE of SUCCESSORS(state)
...
```

What Probabilities to Use?

In expectimax search, we have a probabilistic model of how the opponent (or environment) will behave in any state

Model could be a simple uniform distribution (roll a die) Model could be sophisticated and require a great deal of computation

We have a node for every outcome out of our control: opponent or environment

The model might say that adversarial actions are likely!

For now, assume that for any state we magically have a distribution to assign probabilities to opponent actions and environmental outcomes

Later on, formalize how to model that as Markov Decision Processes

Nondeterministic games in practice

Dice rolls increase b: 21 possible rolls with 2 dice
Backgammon $\approx$ 20 legal moves (can be 6,000 with 1-1 roll)

$$\text{depth } 4 = 20 \times (21 \times 20)^3 \approx 1.2 \times 10^9$$

As depth increases, probability of reaching a given node shrinks
$\Rightarrow$ value of lookahead is diminished

α–β pruning is much less effective

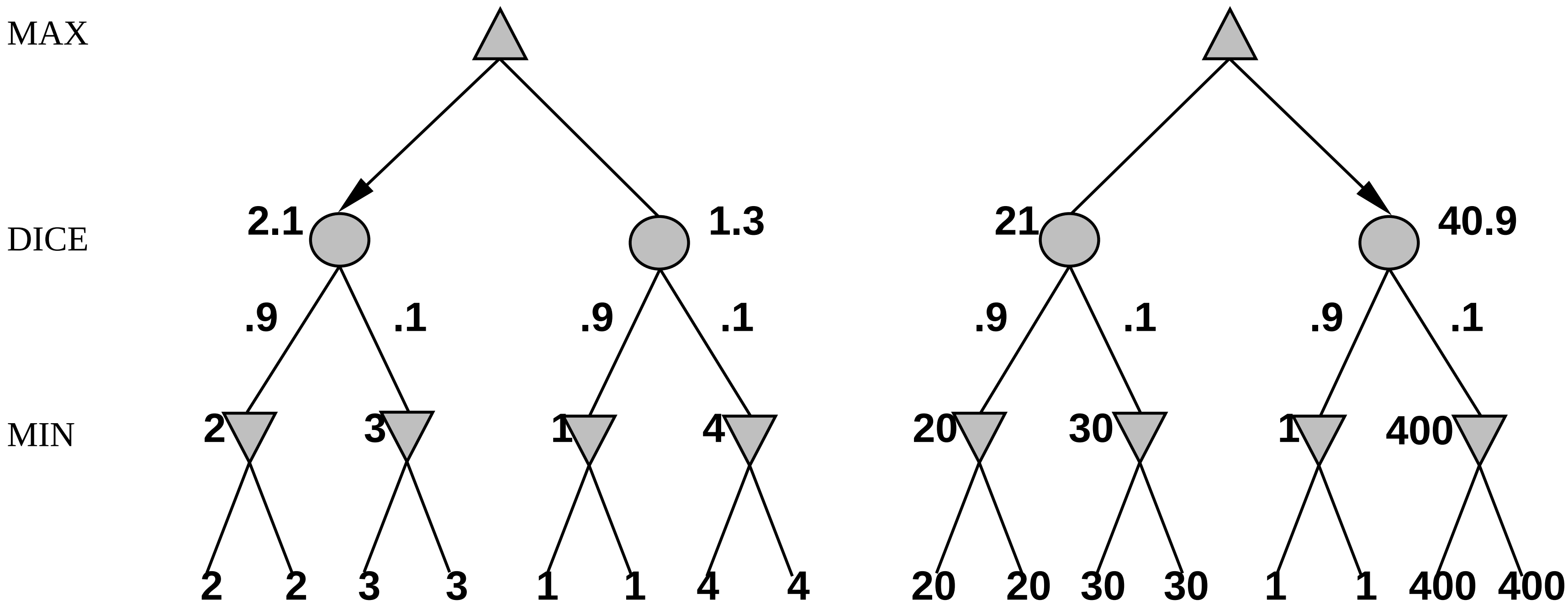
TDGammon uses depth-2 search + very good Eval
$\approx$ world-champion level

What Utilities to Use?

For minimax, terminal function scale doesn't matter - just ordering

For expectimax, we need magnitudes to be meaningful

Digression: Exact values DO matter



Behaviour is preserved only by positive linear transformation of EVAL

Hence EVAL should be proportional to the expected payoff

Reminder: Probabilities

- A random variable represents an event whose outcome is unknown
- A probability distribution is an assignment of weights to outcomes
- Example: traffic on freeway?
  - Random variable: T = whether there's traffic
  - Outcomes: T in none, light, heavy
  - Distribution:
    - P(T=none) = 0.25
    - P(T=light) = 0.55
    - P(T=heavy) = 0.20

Reminder: Probabilities, continued

Some laws of probability (more later):

♢ Probabilities are always non-negative

♢ Probabilities over all possible outcomes sum to one

As we get more evidence, probabilities may change:

♢ P(T=heavy) = 0.20, P(T=heavy — Hour=8am) = 0.60

♢ We'll talk about methods for reasoning and updating probabilities later

Reminder: Expectations

We can define function f(X) of a random variable X

The expected value of a function is its average value, weighted by the probability distribution over inputs

Example: How long to get to the airport?

- Length of driving time as a function of traffic: L(none) = 20, L(light) = 30, L(heavy) = 60
- What is my expected driving time?
  - Notation: E[L(T)]
  - Remember, P(T) = none: 0.25, light: 0.5, heavy: 0.25
  - E[L(T)] = L(none) * P(none) + L(light) * P(light) + L(heavy) * P(heavy)
  - E[L(T)] = (20 * 0.25) + (30 * 0.5) + (60 * 0.25) = 35

Expectimax for Pacman

- Notice that we've gotten away from thinking that the ghosts are trying to minimize pacman's score
- Instead, they are now a part of the environment
- Pacman has a belief (distribution) over how they will act
- Quiz: Can we see minimax as a special case of expectimax?
- Quiz: what would pacman's computation look like if we assumed that the ghosts were doing 1-ply minimax and taking the result 80% of the time, otherwise moving randomly?
- If you take this further, you end up calculating belief distributions over your opponents' belief distributions over your belief distributions, etc...
- Can get unmanageable very quickly!

Games of imperfect information

E.g., card games, where opponent's initial cards are unknown

Typically we can calculate a probability for each possible deal

Seems just like having one big dice roll at the beginning of the game*

Idea: compute the minimax value of each action in each deal,
then choose the action with highest expected value over all deals*

Special case: if an action is optimal for all deals, it's optimal.*

GIB, current best bridge program, approximates this idea by
1) generating 100 deals consistent with bidding information
2) picking the action that wins most tricks on average

Summary

Games are fun to work on! (and dangerous)

They illustrate several important points about AI

$\diamondsuit$ perfection is unattainable $\Rightarrow$ must approximate

$\diamondsuit$ good idea to think about what to think about (metareasoning)

$\diamondsuit$ uncertainty constrains the assignment of values to states

$\diamondsuit$ optimal decisions depend on information state, not real state

Pacman demo and Suicide Pacman Exercise